IMPROVEMENT OF A CONSTITUTIVE MODEL FOR PREDICTING FLOW BEHAVIOR OF NANOCOMPOSITES

Undergraduate Research Thesis

Presented in Partial Fulfillment of the Requirements for Graduation with Research Distinction in the College of Engineering of The Ohio State University

By

Timothy Kremer

William G. Lowery Department of Chemical and Biomolecular Engineering

2013

Thesis Committee:

Dr. Kurt W. Koelling, Advisor

Dr. David L. Tomasko

ABSTRACT

This research project focuses on the development of a constitutive model to predict the flow properties of polymer/nanoparticle composites (nanocomposites). Nanocomposites have gained much interest due to the ability of the nanoparticle to improve properties of the pure polymer such as electrical and thermal conductivities and mechanical strength. The high surface area/volume ratio of the nanosize particles gives these improved properties at small loading levels compared to larger conventional particles. Predicting the flow behavior is important when using the nanocomposite in processes such as spraying, extruding and molding. Two types of experiments were performed. Shear flows at a constant shear rate and small amplitude oscillatory shear flows. These flows were induced on the pure polymer or nanocomposite and the stress recorded as a function of time. Steady shear flows were studied both in the forward and reverse directions with varying rest periods between flow reversals. A constitutive model is used for predicting nanoparticle orientation and flow behavior. There are several parameters in the model that need to be fit to experimental data to accurately predict flow properties of the nanocomposite. Two model parameters were fit to experimental data to give the most accurate prediction of flow behavior. These optimized parameters allow the model to give more accurate predictions of shear viscosity. The model was also expanded to be able to make stress predictions for small amplitude oscillatory shear flows. The predictions from this model can be used to develop and optimize large scale nanocomposite manufacturing processes.

ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr. Kurt Koelling, for all his guidance and advice during the course of this project. Dr. Koelling was always willing to give explanation and insight while also giving challenging assignments. I would also like to thank Dr. Christopher Kagarise and William Murch whose research is the basis for this project. I would again like to thank William Murch for his insight on programming and for all the experimental data that is used in this project.

Vita

2008……..……..Delaware Hayes High School

2013……..….B.S. Chemical and Biomolecular Engineering, The Ohio State University

Publications

J.S. Tumuluru, S. Sokhansanj, C.T. Wright, T.G. Kremer "GC Analysis of Volatiles and Other Products from Biomass Torrefaction Process" Advanced Gas Chromatography-Progress in Agricultural, Biomedical and Industrial Applications, (2012), 211-234

J.S. Tumuluru, T.G. Kremer, C.T. Wright, R.D. Boardman "Proximate and Ultimate Compositional Changes in Corn Stover during Torrefaction using Thermogravimetric Analyzer and Microwaves". 2012 ASABE Annual International Meeting, (2012)

Field of Study

Major Field: Chemical and Biomolecular Engineering

TABLE OF CONTENTS

LIST OF TABLES

TABLE OF FIGURES

INTRODUCTION

Nanoparticles have gained interest as components in composites due to their ability to impart improved properties to a polymer at lower loading levels than traditional, larger particles. This is possible because of the higher surface area of the smaller particles allowing for more contact with the polymer matrix (Jaing *et al*., 2005; Mencke *et al*., 2004). In the past, research efforts focused on using carbon nanotubes as nanoscale additives for nanocomposites due to their small size and ability to impart mechanical strength, electrical conductivity and thermal conductivity to the composite (Dyke and Tour, 2004; Geng *et al*., 2002; Ramasubramaniam *et al*., 2003; Biercuk *et al*., 2002; Choi *et al*., 2003; Du *et al*., 2006; Yang *et al*., 2005). Their implementation has been limited mainly due to the high cost of carbon nanotubes.

Carbon nanofibers (CNFs) have been presented as a cheaper alternative to carbon nanotubes. They are typically around 100 times larger in length and diameter than carbon nanotubes but are approximately 500 times less expensive (Wang *et al.*, 2006). CNFs impart many of the same properties when added to a polymer including mechanical strength, electrical conductivity and thermal conductivity. These properties imparted on the composite by the CNFs, is dependent on the orientation of the CNFs (Miyazono *et al.*, 2011). If the CNFs have a large degree of alignment in a single direction, the conductivities and tensile strength will be larger in that direction compared to a direction perpendicular to the alignment.

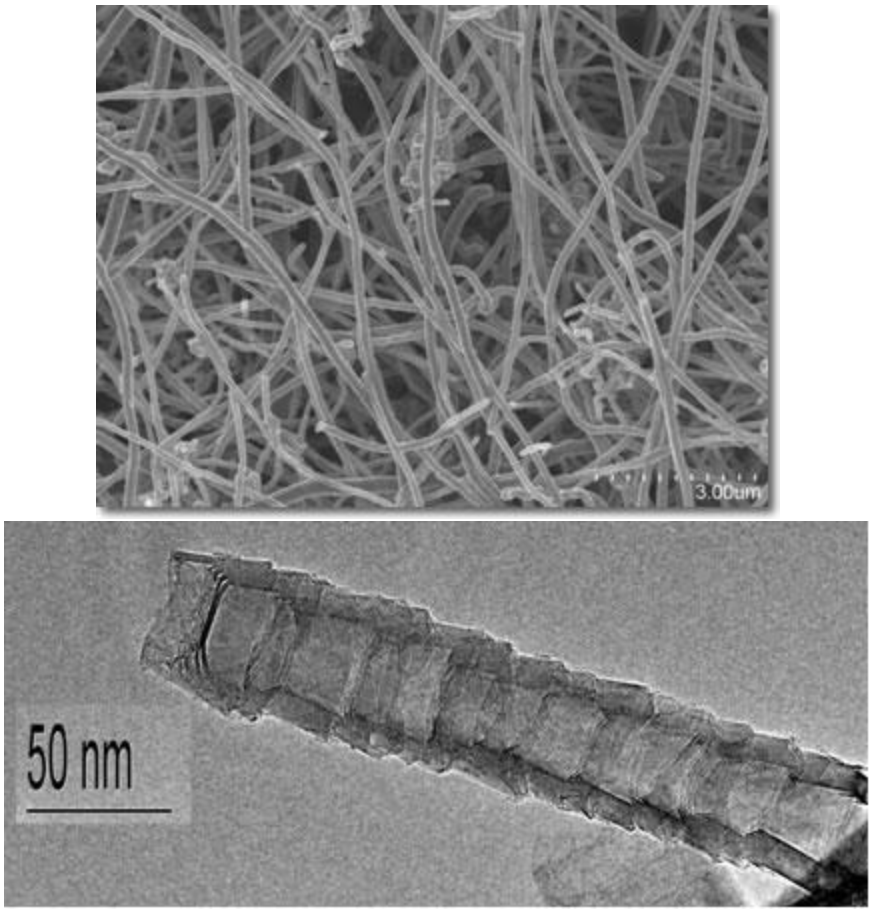


Figure 1: Images of CNFs (AZoNano, 2012)

Applications for CNF composites include possible automotive applications in paintability to lower application cost and waste through electrostatic painting (Figure 2), seals to reduce noise, tires to give improved durability and traction, hoses and belts to lower maintenance, motor mounts to reduce vibration and body panels to reduce weight and give better thermal performance (Hammel et al.).

Possible applications for CNF nanocomposite materials include high capacity, high power anode materials for energy storage. The high electrical conductivity can be tailored to applications in lighting strike protection, antistatic clean rooms and fuel lines, electromagnetic interference (EMI) and radio frequency (RF) shielding (C.L. Lake). Automotive body parts can

be made of these composites to take advantage of electrostatic painting and also be lightweight while retaining high mechanical strength.

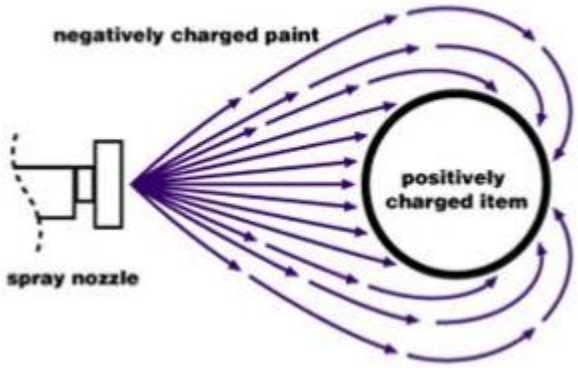


Figure 2: Electrostatic painting process (C.L. Lake)

CNF/polymer composite rheology has been previously studied by Dr. Koelling's research group focusing on shear and extensional rheology and the development of a constitutive model for use in predicting stresses in the composite and CNF orientation. Other groups have also performed research on the shear rheology of CNF/polymer composites. Studies have included the use of different polymers such as polypropylene (Carneiro and Maia, 2000; Lozano et al., 2001; Ceccia et al., 2008; Cortés et al., 2002), polyethylene (Lozano et al., 2004), poly(ether ether ketone) (Modi et al., 2010) and poly(methyl methacrylate) (Varela-Rizo et al., 2011; Varela-Rizo et al., 2012). The storage and loss modulus have been shown to increase with the addition of CNFs (Carneiro and Maia, 2000; Lozano et al., 2001; Ceccia et al., 2008; Cortés et al., 2002; Lozano et al., 2004; Modi et al., 2010; Varela-Rizo et al., 2011; Varela-Rizo et al., 2012). In general, the steady-state viscosity has also been shown to increase with the incorporation of CNFs (Carneiro and Maia, 2000; Ceccia et al., 2008; Ma et al., 2003). However, one group's studies on a polycarbonate/CNF composite have shown a decrease in steady-state

viscosity for a forward shear when CNFs are added to the polymer. These results were attributed to a possible poor fiber-matrix adhesion leading to possible layered flow (Caldeira et al., 1998; Carneiro et al. 1998).

This study's main focus is on improving the accuracy with which a constitutive model, previously studied by Dr. Koelling's group, can predict transient viscosity in forward and reverse shear flows (Kagarise et al., 2008; Kagarise et al., 2010; Kagarise et al., 2011; Miyazono et al., 2011; Xu et al., 2005). This model also has the capability to predict CNF orientation and normal stress. Reverse flows are performed by first shearing the nanocomposite in one direction. Then a rest time takes place where the nanocomposite relaxes while no shear force is applied. The sample is then sheared in the "reverse" direction when compared to the original shear. Figure 3 shows the general evolution of shear rate through a forward, rest time and then reverse experiment. The varying amount of rest time given allows the polymer chain to relax and the model's ability to predict this relaxation can be tested. Reverse flows are also important because previous work by this group has suggested that the nanoparticles stay in their aligned position after the initial shear flow and do not reorient. This means the model can be tested in a different situation where there is little reorientation of the CNFs and thus there is little additional stress being contributed by changes in orientation.

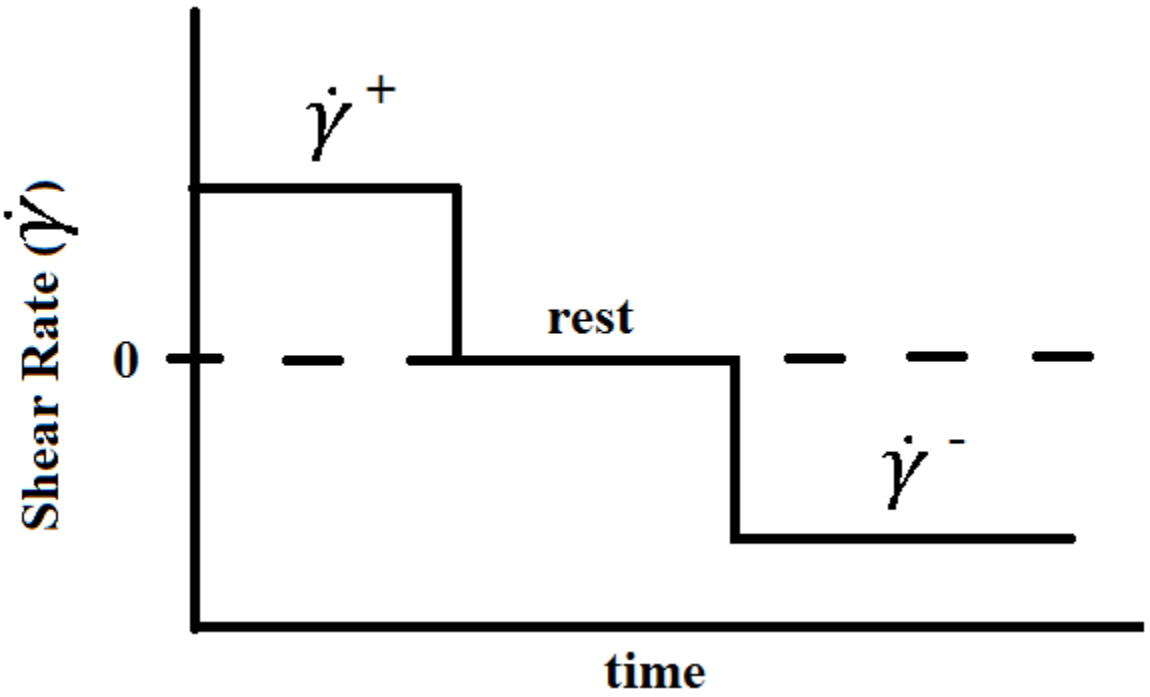


Figure 3: Shear rate during an experiment with a forward flow ($\dot{\gamma}^+$), rest period and reverse flow ($\dot{\gamma}^-$)

The accuracy improvement is achieved by finding optimized model parameters that minimize the error between model predictions and experimental data for these flows. The experimental data used in the fitting consisted of transient viscosity data from trials at two different CNF loadings, 2wt% and 5wt%, two shear rates, 0.1 s^{-1} and 1 s^{-1} and three different rest periods, 0 s, 20 s and 400 s, in between forward and reverse flows.

The model was also expanded to make stress predictions in small amplitude oscillatory shear (SAOS) flows, a flow regime that has not been previously investigated by the proposed model. SAOS experiments are performed by shearing the sample in one direction until some specified strain is reached and then the shear direction is reversed until the specified strain is reached again and the process repeats giving an oscillatory strain in the sample with respect to time with a frequency ω. SAOS experiments were performed on the pure polymer, 2wt% and 5wt% composites with a range of frequencies from 0.015 to 100 radians/s. The experimental

determination for the storage (G') and loss (G") modulus are then compared to model predictions for these values to assess the model's ability to predict these types of flows.

MATERIALS & METHODOLOGY

The polymer used in this study to combine with the CNFs was polystyrene (PS). The polystyrene used in this study was supplied by the Chevron Phillips Chemical Co. LP (MC3600). The PS has a specific gravity of 1.03 and a melt flow rate of 13.0 g/10 min (200°C/5 kg, method: ASTM D1238).

The CNFs used in this study were pyrograf III (type PR-24-XT-HHT), manufactured by Applied Sciences, Inc, and used as-received. These are vapor-grown carbon nanofibers and have a stacked cup structure as shown in Figure 4. These as-produced fibers had lengths of approximately 50-200 μm and average diameters of 100 nm.

Figure 4: Stacked cup structure of Pyrograf III CNFs

The nanocomposite material was created by combining the PS and the CNFs in a DACA Instruments twin-screw microcompounder at 200°C with a screw rotation rate of 250 RPM and mixing for 4 minutes. From the microcompounder, the composite samples were extruded from a 2 mm die, and after cooling, were cut into ~2-3 mm long pellets. Pure PS was also tested in this study so the PS used in experiments also underwent this microcompounder and extrusion process to account for any degradation that may occur during the process. The processing conditions used here were chosen to balance good dispersion of the nanofibers with limited degradation of the polymer.

The resulting pellets were then compression molded into 25 mm diameter disks of 0.9-1.2 mm thickness. This was achieved by placing the pellets into molds and melting them for 15 minutes at 200 °C. This was followed by quickly compressing and decompressing the samples four times to eliminate air bubbles. Pressure was then reapplied and held for 10 minutes. The samples were then allowed to cool and pressure released to then remove the sample from the mold. The samples were then placed in a vacuum oven at 70 °C for at least 24 hours and remained there until testing to avoid contact with air and absorption of moisture.

The shear rheology of pure PS and the nanocomposites was experimentally tested using a TA Instruments strain-controlled rheometer (ARES LS2) with a torque transducer (0.02-2000 g-cm) and a normal force transducer (2-2000 g). Parallel plates with 25 mm diameters with a gap distance of 0.9-1.2 mm were used for all experiments. The samples were allowed to rest at the test temperature of 160 °C for 15 minutes prior to testing in order to reach thermal equilibrium and relax any stresses obtained previous to testing. This rheometer and procedure were used for both SAOS and transient shear experiments.

The SAOS tests were performed pure polymer and nanocomposite samples of 2 and 5wt% at frequencies between 0.016-100 radians/second with logarithmic spacing. The transient shear experiments were explored at several factors as shown in Table 1 in a full factorial design. Data was recorded in the transient experiments at logarithmically spaced intervals. The total experiment time was dependent on the shear rate with a 500 second time being used for a single flow direction at 0.1 s^{-1} and 50 seconds being used at 1 s^{-1}.

Table 1: Factors and levels in transient shear experiments

Factor	**Levels**
CNF Loading	0, 2 and 5wt%
Shear Rate	0.1 s^{-1} and 1 s^{-1}
Direction of Flow	Forward and Reverse
Rest Period in between Forward and Reverse	0 s, 20 s and 400 s

All experimental data presented in this project was collected by William Murch and not the author.

CONSTITUTIVE MODEL

This study uses a constitutive model to predict viscosity and nanoparticle orientation in the composite during flows. The model has been previously studied and validated for systems with carbon nanofibers and nanoclays (Kagarise, 2009). Both shear and extensional flows have been previously studied as well as the effect of nanofiber orientation by studying reverse flows (Murch, 2011). This model has several parameters that have not been well studied and finding optimum values for these parameters can improve model accuracy. The model is based on the Gieskus Model and modified to incorporate the effects of nanoparticles in the polymer. The

model has four governing equations the together predict the overall composite stress [Eq. (1)] and the average orientation of the nanoparticles [Eq. (4)].

$$\tau_{ij}^{c} = -p\delta_{ij} + 2\eta_s D_{ij} + \tau_{ij}^{p} + \tau_{ij}^{CNF}, \tag{1}$$

$$\sigma\tau_{ij,m}^{p} + \lambda_m \frac{D\tau_{ij,m}^{p}}{Dt} + \frac{\alpha_m\lambda_m}{\eta_{p,m}}\left(\tau_{ik,m}^{p}\tau_{kj,m}^{p}\right) + \frac{3(1-\sigma)}{2}\left(a_{ik}\tau_{kj}^{p} + \tau_{ik}^{p}a_{kj}\right) = 2\eta_{p,m}D_{ij}, \tag{2}$$

$$\tau_{ij}^{CNF} = 2[\eta_s + \eta]\Phi\left[AD_{kl}a_{ijkl} + B\left(D_{ik}a_{kj} + a_{ik}D_{kj}\right) + CD_{ij} + 2Fa_{ij}D_r\right], \tag{3}$$

$$\frac{da_{ij}}{dt} = \left(W_{ik}a_{kj} - a_{ik}W_{kj}\right) + \chi\left(D_{ik}a_{kj} + a_{ik}D_{kj} - 2D_{kl}a_{ijkl}\right) + 4C_I\Pi_D^{\frac{1}{2}}(\delta_{ij} - 3a_{ij}). \tag{4}$$

Equation 1 gives the total stress of the composite as the sum of the stress from the polymer, τ_{ij}^{p}, nanoparticles, τ_{ij}^{CNF}, the pressure maintaining incompressibility, p, and the Newtonian solvent ($2\eta_s D_{ij}$), where η_s is the solvent viscosity and D_{ij} is the symmetric part of Eulerian velocity gradient (Azaiez, 1996). In the system used in this study, no solvent is present.

Equation 2 is the modified multi-mode Gieskus model that predicts the polymer stress contribution (Bird et al., 1987). The total polymer stress contribution is a summation from all modes:

$$\tau_{ij}^{p} = \sum_{m=1}^{N} \tau_{ij,m}^{p} \tag{5}$$

The upper convected time derivative of τ_{ij}^{p} is

$$\frac{D\tau_{ij}^{p}}{Dt} = \frac{d}{dt}\tau_{ij}^{p} - W_{ik}\tau_{kj}^{p} + \tau_{ik}^{p}W_{kj} - D_{ik}\tau_{kj}^{p} - \tau_{ik}^{p}D_{kj}. \tag{6}$$

Where W_{ij} is the skew portion of the Eulerian velocity gradient. For shear flows,

$$D_{ij} = \begin{bmatrix} 0 & \dot{\gamma}/2 & 0 \\ \dot{\gamma}/2 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \text{ and } W_{ij} = \begin{bmatrix} 0 & \dot{\gamma}/2 & 0 \\ -\dot{\gamma}/2 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}. \quad (7)$$

The stress tensor is symmetric with six terms (τ_{11}, τ_{22}, τ_{33}, $\tau_{12} = \tau_{21}$, $\tau_{13} = \tau_{31}$, and $\tau_{23} = \tau_{32}$), only four of which are nonzero in shear flow (τ_{11}, τ_{22}, τ_{33}, and τ_{12}). The axes numbering with reference to a shear flow is shown in Figure 5.

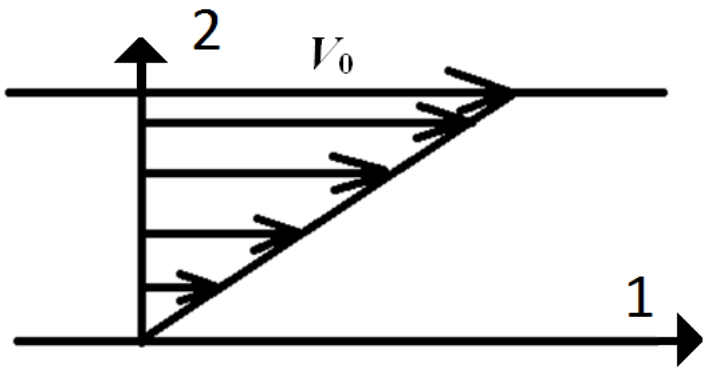


Figure 5: A shear flow with the top plate moving at a velocity v_0 and the axes numbered as in the model equations

Each of the four nonzero tensor components has a resulting equation. This creates four coupled, non-linear differential equations that must be solved simultaneously. These tensor-component equations simplify for each mode to:

$$\frac{d}{dt}\tau_{11}^p = \frac{-\sigma*\tau_{11}^p - 3(1-\sigma)(\tau_{11}^p a_{11} + \tau_{12}^p a_{12})}{\lambda} - \frac{\alpha}{\eta_p}\left(\tau_{11}^2 + \tau_{12}^2\right) + 2\dot{\gamma}\tau_{12}^p \quad (8)$$

$$\frac{d}{dt}\tau_{22}^p = \frac{-\sigma*\tau_{22}^p - 3(1-\sigma)(\tau_{12}^p a_{12} + \tau_{22}^p a_{22})}{\lambda} - \frac{\alpha}{\eta_p}\left(\tau_{12}^2 + \tau_{22}^2\right) \quad (9)$$

$$\frac{d}{dt}\tau_{33}^p = \frac{-\sigma*\tau_{33}^p - 3(1-\sigma)(\tau_{33}^p a_{33})}{\lambda} - \frac{\alpha}{\eta_p}\left(\tau_{33}^2\right) \quad (10)$$

$$\frac{d}{dt}\tau_{12}^p = \frac{\dot{\gamma}\eta_p}{\lambda} - \frac{\sigma\tau_{12}^p}{\lambda}\frac{-\sigma*\tau_{11}^p - 3(1-\sigma)(\tau_{12}^p a_{11} + \tau_{22}^p a_{12} + \tau_{12}^p a_{22})}{\lambda} - \frac{\alpha}{\eta_p}\left(\tau_{11}^p\tau_{12}^p + \tau_{22}^p\tau_{12}^p\right) + \dot{\gamma}\tau_{22}^p \quad (11)$$

Also from Equation (2), η_p, λ, and α represent the polymer zero-shear viscosity, relaxation time, and mobility factor, respectively, for the melt-phase polymer matrix. These three parameters are fitting parameters that are only dependent on pure polymer behavior. The orientation tensor, a, describes the average orientation of nanoparticles in the composite. This tensor will be described in more detail later in this section. The parameter σ is the polymer-particle interaction parameter which accounts for physical interactions between the nanoparticle and the polymer. In previous studies with this model, σ has always been assumed to be equal to one, giving no interactions between the nanoparticles and the polymer. This assumption also makes the term with the orientation tensor, a, drop out of the equation. This effectively decouples the polymer stress equation, equation 2, and the orientation evolution equation, equation 4, and allows the two equations to be solved separately. This study will attempt to find an optimum value for σ to help make model predictions that coincide more closely with experimental measurements. In doing so, the equations become more difficult to solve due to the necessity to solve equations 2 and 4 simultaneously.

The flow-induced stress associated with the presence of carbon nanofibers is described by Equation (3) (Tucker, 1991). The term η is the viscosity contribution from the polymer, which is defined for shear flow as

$$\eta = \frac{\tau_{12}^{p}}{\dot{\gamma}}, \tag{12}$$

Where τ_{12}^{p} is the polymer stress in the 1-2 direction. The parameter ϕ is the volume fraction of CNFs in the composite which is calculated using the mass fraction of CNFs added the CNF density and the polymer density. The parameters *A, B, C and F* are nanoparticle shape factors and are functions of the volume fraction, ϕ, and aspect ratio, r=$\frac{Length\ of\ CNF}{Diameter\ of\ CNF}$. The aspect ratio for this

system has been previously experimentally determined to be $r = 33$ (Kagarise *et al.*, 2010; Kagarise *et al.*, 2011). Definitions for these shape factors have been proposed (Tucker, 1991; Dinh and Armstrong, 1984; Shaqfeh and Fredrickson, 1990) for different concentration regimes (dilute or semidilute), states of nanoparticle orientation (isotropic or aligned), and flow conditions (transient or steady state). The system in this study falls under the semi-dilute and aligned regime for which the shape factors *B*, *C*, and *F* are equal to zero, and the only nonzero shape factor is *A*. This group has previously analyzed the available options from these previous works and has determined the best relation for the shape factor *A* for this CNF/PS system to be (Kagarise *et al.*, 2010; Kagarise *et al.*, 2011):

$$A = \frac{r^2}{3\ln(\frac{\sqrt{\pi}}{\sqrt{\phi}})} \tag{13}$$

Where r is the average aspect ratio. For samples prepared by the same procedure the aspect ratio was found to be $r = 33$ (Kagarise *et al.*, 2010; Kagarise *et al.*, 2011). The term D_r is the rotary diffusivity due to Brownian motion, which is defined as $D_r = 2C_I\Pi_D^{1/2}$. Π_D is the second invariant of D_{ij}, the symmetric part of Eulerian velocity gradient. C_I is the particle-particle interaction parameter, otherwise known as the Folgar-Tucker constant (Folgar and Tucker, 1984). There has been minimal effort previously in attempting to fit C_I quantitatively to experimental data. An optimized value for this parameter will be found in this study to further increase the accuracy of the model. After substituting for the shape factors and the closure approximation for the fourth order orientation tensor, the individual components of the CNF stress tensor are given by:

$$\tau_{12}^{CNF} = 2\eta\phi\left[A\dot{\gamma}\left[\begin{matrix}(27a_{11}a_{22}a_{33} - 27a_{33}a_{12}^2)\left(-\frac{1}{35} + \frac{(a_{11}+a_{22})}{7}\right)\\ +(a_{12}^2)(1 - 27a_{11}a_{22}a_{33} + 27a_{33}a_{12}^2)\end{matrix}\right]\right] \tag{14}$$

$$\tau_{11}^{CNF} = 2\eta\phi\left[A\dot{\gamma}\left[(27a_{11}a_{22}a_{33} - 27a_{33}a_{12}^2)\left(\frac{3a_{12}}{7}\right) + (a_{11}a_{12})(1 - 27a_{11}a_{22}a_{33} + 27a_{33}a_{12}^2)\right]\right] \tag{15}$$

$$\tau_{22}^{CNF} = 2\eta\phi\left[A\dot{\gamma}\left[(27a_{11}a_{22}a_{33} - 27a_{33}a_{12}^2)\left(\frac{3a_{12}}{7}\right) + (a_{22}a_{12})(1 - 27a_{11}a_{22}a_{33} + 27a_{33}a_{12}^2)\right]\right] \quad (16)$$

$$\tau_{33}^{CNF} = \left(2\eta\phi\left[A\dot{\gamma}\left[(27a_{11}a_{22}a_{33} - 27a_{33}a_{12}^2)\left(\frac{a_{12}}{7}\right) + (a_{33}a_{12})(1 - 27a_{11}a_{22}a_{33} + 27a_{33}a_{12}^2)\right]\right]\right) \quad (17)$$

The orientation evolution equation is shown in Equation (4). This equation predicts the CNF orientation in the polymer matrix through the orientation tensor, a. To solve for the orientation tensor, first consider a vector, **p**, which describes the orientation of a single fiber in the polymer matrix. This vector can be expressed using two angles, θ and ϕ, as seen in Figure 6.

$$\mathbf{p}=(p_1,p_2,p_3)=(\cos\theta,\sin\theta\sin\phi,\sin\theta\cos\phi) \quad (18)$$

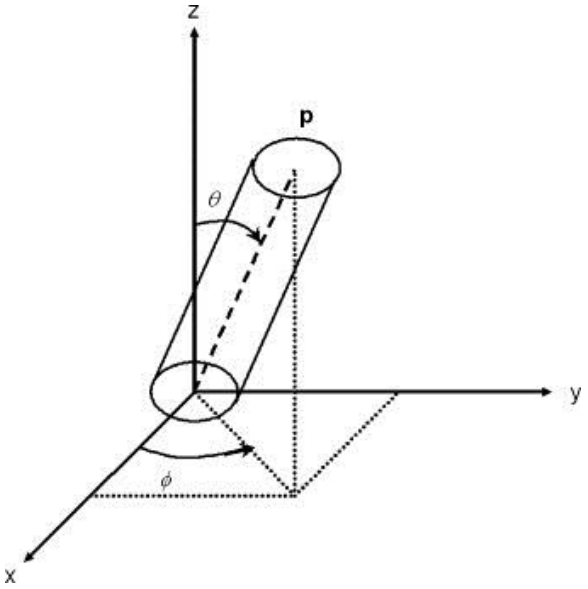


Figure 6: Fiber Orientation Vector, p, Described by Angles θ and ϕ (Kagarise et al., 2011)

In real systems with many fibers the average orientation of the fibers must be found. This is done using a probability distribution function, ψ(θ,ϕ). The probability distribution function describes the probability, P, of finding a fiber between angles θ_1 and θ_2 and ϕ_1 and ϕ_2 using the integral:

$$P = \int_{\phi 1}^{\phi 2}\int_{\theta 1}^{\theta 2}\psi(\theta,\phi)\sin\theta d\theta d\phi \quad (19)$$

The probability distribution function can also be defined as a function of the fiber orientation vector **p** as shown in figure 1. The probability distribution function can be written as $\psi(\mathbf{p})$. The second and fourth order orientation tensors can then be defined as the dyadic product of the fiber orientation vector, **p**, with itself and averaged over orientation space (Advani and Tucker, 1987).

$$a_{ij}=\int p_i p_j \psi(\mathbf{p}) d\mathbf{p} \tag{20}$$

$$a_{ijkl}=\int p_i p_j p_k p_l \psi(\mathbf{p}) d\mathbf{p} \tag{21}$$

In terms of averaged angles this gives the second-order orientation tensor as:

a=

$$\begin{pmatrix} \cos^2\theta & \sin\theta\cos\theta\sin\phi & \sin\theta\cos\theta\cos\phi \\ \sin\theta\cos\theta\sin\phi & \sin^2\phi\sin^2\theta & \sin^2\theta\sin\phi\cos\phi \\ \sin\theta\cos\theta\cos\phi & \sin^2\theta\sin\phi\cos\phi & \cos^2\phi\sin^2\theta \end{pmatrix} \tag{22}$$

The a_{ij} components of importance for a shear flow are:

$$a_{11} = \cos^2\theta \tag{23}$$

$$a_{12} = \sin\theta\cos\theta\sin\phi \tag{24}$$

$$a_{22} = \sin^2\phi\sin^2\theta \tag{25}$$

$$a_{33} = \cos^2\phi\sin^2\theta \tag{26}$$

The fourth-order orientation tensor a_{ijkl} cannot be solved for directly and instead must be related to the second-order tensor a_{ij}, using a closure approximation. Popular closure approximations include a linear approximation [Equation (27)], which works better for randomly-aligned states, the quadratic approximation [Equation (28)], which works better for highly-aligned states, and the hybrid closure approximation [Equation (29)], which is a combination of the linear and quadratic approximations and works well across the range of orientations (Advani and Tucker, 1990). Previous work from this group has investigated all three closure approximations and shown that the hybrid closure approximation

provides the best predictions for these PS/CNF systems (Wang et al., 2006). The hybrid closure approximation, shown in Equation (29), was used in this study.

$$\hat{a}_{ijkl} = -\frac{1}{35}\left(\delta_{ij}\delta_{kl} + \delta_{ik}\delta_{jl} + \delta_{il}\delta_{jk}\right) + \frac{1}{7}(a_{ij}\delta_{kl} + a_{ik}\delta_{jl} + a_{kl}\delta_{ij} + a_{jl}\delta_{ik} + a_{jk}\delta_{il} + a_{il}\delta_{jk}), \quad (27)$$

$$\tilde{a}_{ijkl} = a_{ij}a_{kl}, \quad (28)$$

$$a_{ijkl} = (1-f)\hat{a}_{ijkl} + f\tilde{a}_{ijkl}, \quad \text{with } f = 1 - 27\det(a_{ij}). \quad (29)$$

The parameter χ is a function of the aspect ratio of the CNFs and is given by equation (26):

$$\chi = \frac{r^2+1}{r^2-1}. \quad (30)$$

Once equations 1-4 have been solved, the total composite stress components can be used to calculate the composite viscosity, first and second normal stress difference. These are shown for shear flows respectively in Equations (31), (32) and (33). The normal stress differences can then be used to calculate the normal force. For a parallel plate configuration, as was used in the experiments for this project, the normal force is the difference between the first and second normal stress difference as seen in Equation (34).

$$\eta_c^+ = \frac{\tau_{12}^c}{\dot{\gamma}} \quad (31)$$

$$N1 = \tau_{11}^c - \tau_{22}^c, \quad (32)$$

$$N2 = \tau_{22}^c - \tau_{33}^c. \quad (33)$$

$$Normal\ Force = N1 - N2 = \tau_{11}^c - 2\tau_{22}^c + \tau_{33}^c \quad (34)$$

SOLVING THE MODEL EQUATIONS

Equations (2) and (4) give rise to coupled, nonlinear differential equation that are highly unlikely to have an analytical solution. These equations must be solved by numerically integrating. The platform used to do that in this study was MATLAB. A sample program that is capable of solving these equations is given in Appendix A. The numerical integration in this case is not straightforward and has caused some complications in converging to correct solutions. Under certain values of parameters the solver fails and the viscosity prediction diverges to negative infinity as shown in Figure 7.

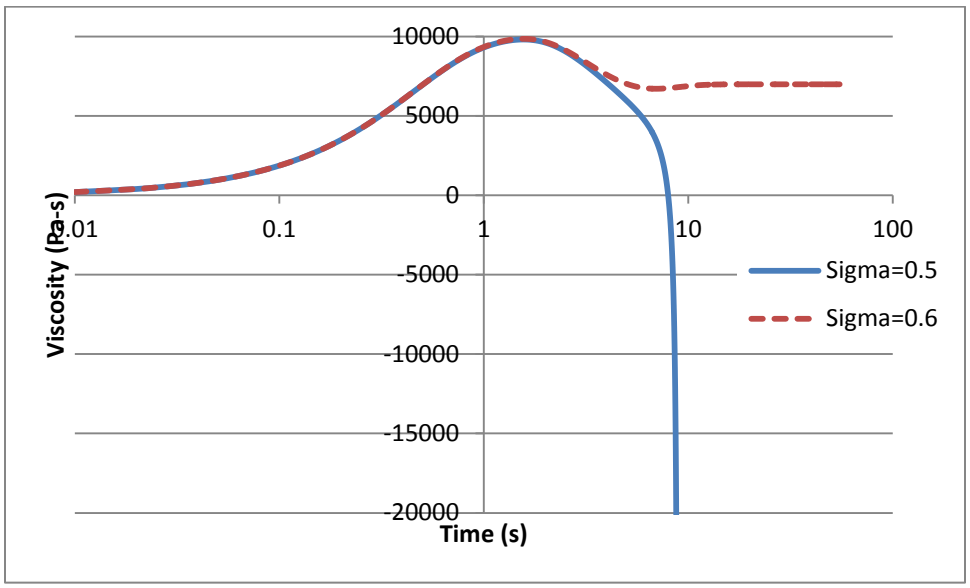


Figure 7: Model prediction diverging or converging to a steady-state value depending on the value of sigma. Both predictions are for 5wt% samples at a shear rate of 1 s^{-1}.

This phenomenon only occurs when sigma does not equal one and the polymer stress equation, Equation, (2), becomes coupled with the orientation evolution equation, Equation (4). Previous studies have always assumed sigma to be equal to one so this phenomenon was never encountered.

For these cases it is possible to solve for the steady-state viscosity algebraically. The result is a reasonable and positive solution for the steady-state viscosity while the transient numerical solver diverges. This would suggest that the equations that are being solved are stiff equations and the numerical

solver diverges from the true solution. In an attempt to fix the problem, all the MATLAB supplied stiff ordinary differential equation (ODE) solvers were tried with little change in result. The Jacobian matrix was solved for and supplied to the ODE solvers with little change. This divergence happens very abruptly with change in parameters and either matches the algebraic steady-state solution or diverges. Problems arise specifically with decreasing sigma value, increasing CNF loading, increasing shear rate, increasing lambda, increasing alpha and increasing aspect ratio of CNFs. If the aspect ratio is decreased to a value less than one the simulation converges for all other parameter values. An aspect ratio less than one corresponds to a negative value of CHI while an aspect ratio greater than one gives a CHI value that is positive. For this study the aspect ratio is defined by the CNFs used in experiments.

Lambda and alpha values are confined to a specific mode of the model. To find the parameter area where the solution matches steady-state predictions, it is only necessary to look at lambda and alpha values only within a single mode. Arbitrarily choosing a minimum sigma value equal to 0.6 and choosing the largest shear rate and CNF loading used in experiments gives rise to Figure 8 that shows the values of lambda and alpha for which the solution converges with all other parameters set to values used in this study that caused the most trouble in converging to a solution.

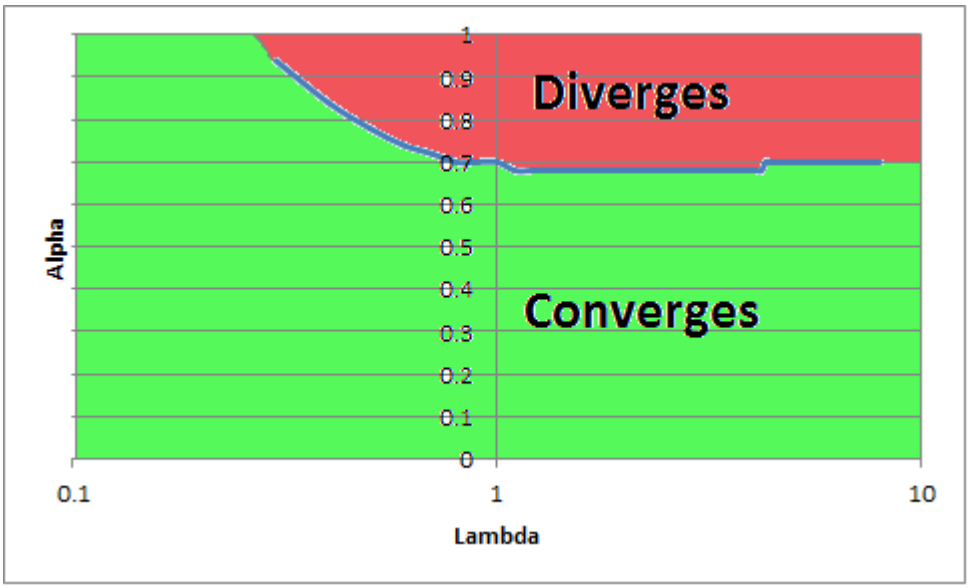
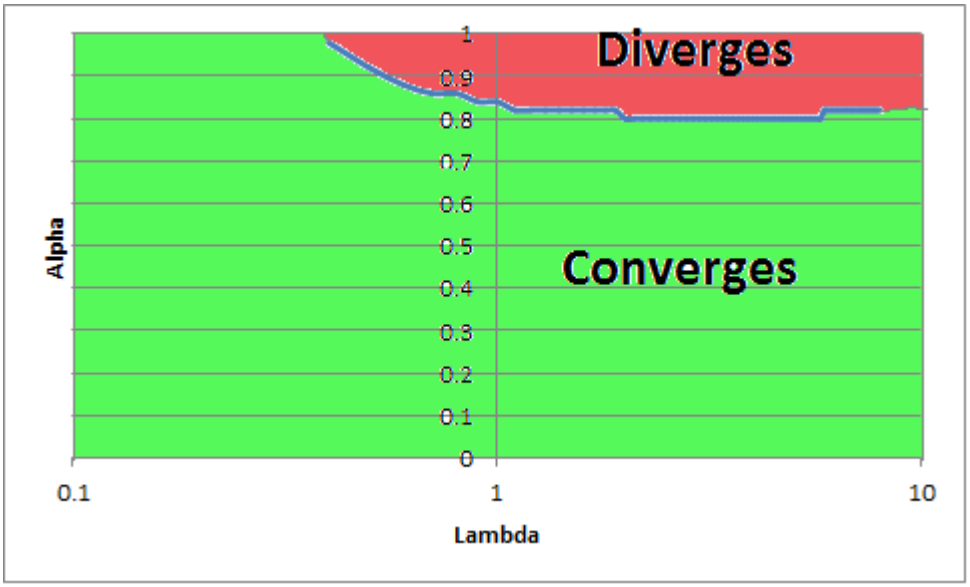


Figure 8: Convergence area for specific combinations of lambda and alpha, with sigma=0.6, 5wt% CNFs, shear rate=1 s^{-1} using the hybrid closure approximation.

An alternative hypothesis for what causes this phenomenon is the closure approximation used to solve for the fourth-order orientation tensor. This tensor cannot be solved for exactly and this error from approximation could cause the failure in model prediction or a discontinuity between the starting time and reaching the steady-state solution. The change in closure approximation does change the range of parameters in which the solution converges as shown in Figure 9 for two alternative closure approximations, the quadratic closure approximation and the linear closure approximation. Figure 8 depicts convergence results for the hybrid closure approximation that is used in this study. While there is a smaller area where the model converges with this closure method it is still used because of its ability to give more accurate predictions for both aligned and nonaligned composite systems when compared to the other two depicted closure methods.



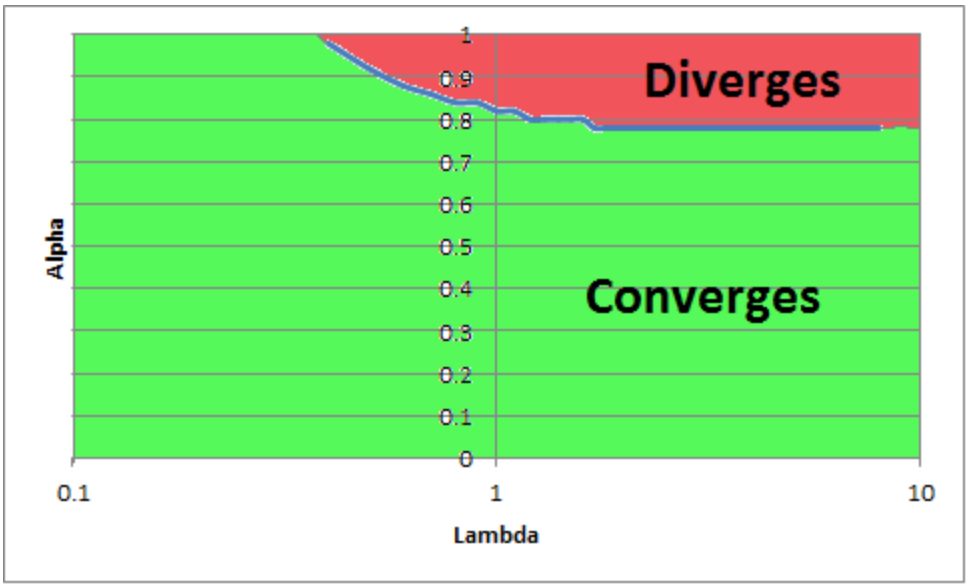


Figure 9: Convergence area for specific combinations of lambda and alpha, with sigma=0.6, 5wt% CNFs, shear rate=1 s^{-1} for a.) Quadratic closure approximation. b.) Linear closure approximation

RESULTS AND DISCUSSION

Pure Polymer Model Fitting

Previous work from this group has found using a Gieskus model with five modes to be an optimum number to give accurate results with a minimum number of parameters needed (Kagarise, 2009). This study will continue to use a five mode model. The first fitting that needed to be done was that for the pure polymer. The parameters that need to be fit for the pure polymer

include the polymer viscosity, $\eta_{p,m}$, the relaxation time, λ_m, and the mobility factor, α_m. The goal of this project was to account for the effects of the addition of the carbon nanofibers into the polymer. This makes the fit to transient polymer viscosity important so that the values of the composite specific parameters are accounting for the change in viscosity due to the addition of CNFs rather than correcting for error already present from the pure polymer predictions. For this reason the polymer parameters are fit both to their analytical solution based on small amplitude oscillatory shear (SAOS) experiments and to the transient viscosity data for the pure polymer.

The values for η_p and λ can be found by performing SAOS experiments and determining experimental values of the shear storage and loss modulus, G' and G" respectively. The Gieskus model has an analytical solution for the G' and G" shown in equations (35) and (36) where ω is the frequency in radians/second.

$$G'(\omega) = \sum_{m=1}^{N} \left(\frac{\eta_{p,m} \lambda_m \omega^2}{1+(\lambda_m \omega)^2} \right), \tag{35}$$

$$G''(\omega) = \sum_{m=1}^{N} \left(\frac{\eta_{p,m} \omega}{1+(\lambda_m \omega)^2} \right), \tag{36}$$

The error between the experimental values and the analytical solution is calculated according to equation (37) and is dependent on the values of η_p and λ.

$$E_{SAOS} = \sum_{i=1}^{N} \left\{ \left[\log_{10} G'_{exp}(\omega_i) - \log_{10} G'(\omega_i) \right]^2 + \left[\log_{10} G''_{exp}(\omega_i) - \log_{10} G''(\omega_i) \right]^2 \right\} \tag{37}$$

The error calculation from the transient shear viscosity experimental data is calculated by solving equations (1)-(4) to attain the model prediction for the viscosity at both shear rates, 0.1 and 1 s^{-1}. Each experimental data point was matched with a model prediction value and error was

calculated according to Equation (38) where η^+ is the transient shear viscosity, t is the time, N is the number of points per experiment and M is the number of transient experiments. Only forward shear experiments were used in the pure polymer fitting giving two total transient viscosity experiments. The transient viscosity prediction is a function of η_p, λ and α. SAOS testing is not dependent on α meaning α is only fit by the transient viscosity experiments.

$$E_{TRANSIENT} = \sum_{j=1}^{M} \sum_{i=1}^{N} \left[\eta^+_{exp}(t_i) - \eta^+_{model}(t_i)\right]^2 \quad (38)$$

The error from both the SAOS experiments and the transient viscosity experiments are combined to some total error after assigning relative weighting to each by dividing it by some value to scale it and then summing the results. A MATLAB program was written to optimize the pure polymer parameters, η_p, λ and α, by minimizing this total error. This program solves the necessary equations to obtain model predictions and calculates the individual errors according to equations (36) and (37). These errors are then scaled to give some relative weighting and then summed to give the total error that is to be minimized. The program iterates this process according to the interior-point algorithm in order to minimize the total error. This program can be seen in Appendix B.

The values that the transient and SAOS errors are divided by for scaling and weighting were adjusted to find a weighting that gave the best possible agreement to the transient viscosity data while still giving a good fit to the G' and G" data. The final scaling factors were chosen as 1.1×10^8 for the transient error and 0.8 for the G' and G" error. The relative magnitude of these factors gives no insight into the percent weighting given to each as there was a large difference in the number of points and the magnitudes of the values being compared for each type of error.

Table 2 shows the final values for η_p, λ and α. The final fitting results to the pure polymer transient viscosity data and the G' and G" data are shown in Figures 10 and 11 respectively.

Table 2: Final values for η_p, λ and α as a result of fitting to pure polymer transient viscosity and G' and G" data.

Mode	**1**	**2**	**3**	**4**	**5**
$\boldsymbol{\eta_p}$	1100.047	2920.194	9277.693	14316.48	7408.915
$\boldsymbol{\lambda}$	0.012288	0.117283	0.74383	2.689761	15.39199
$\boldsymbol{\alpha}$	0.504132	0.678743	0.680016	0.317377	0.279756

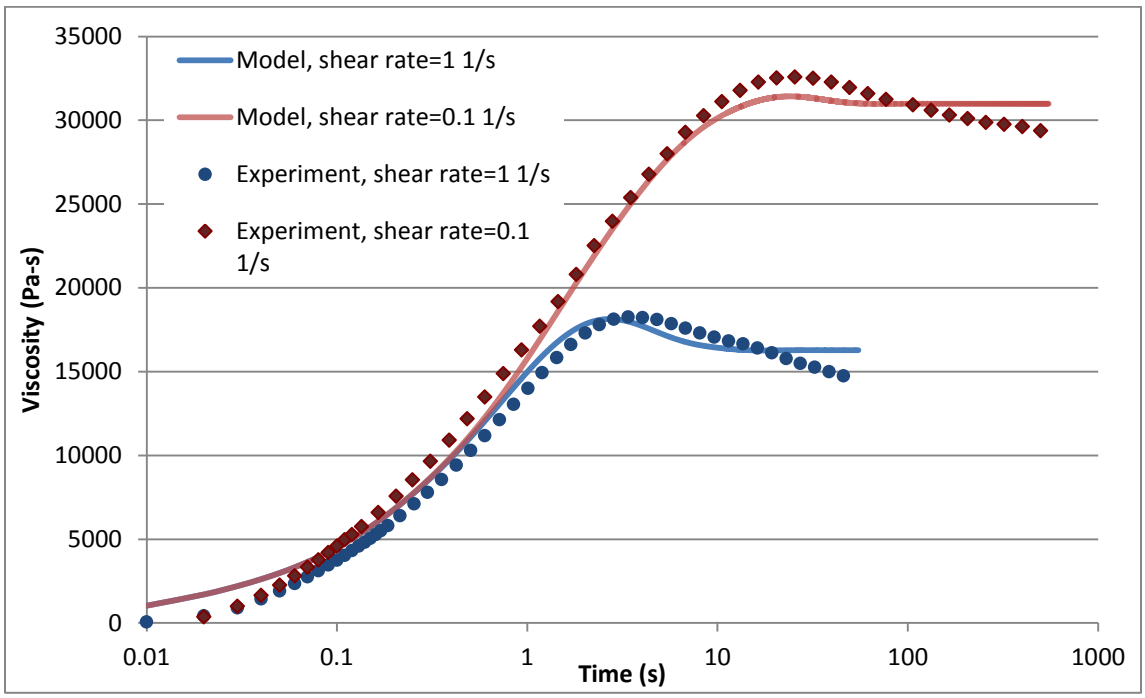


Figure 10: Model predictions and experimental results for shear flows of the pure polymer

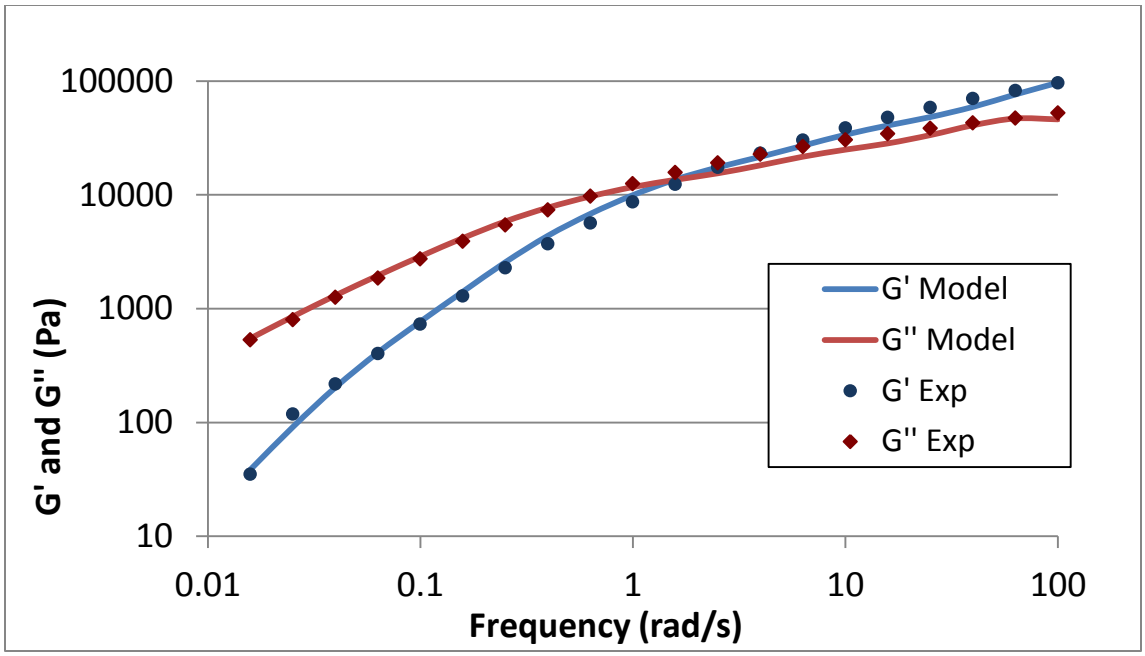


Figure 11: Analytical solution and experimental results for G' and G" from SAOS experiments

The model predictions show close agreement to both data sets. This accuracy for polymer predictions will help assure that when fitting parameters to the nanocomposite viscosity data, they will be accounting for changes caused by the addition of the CNFs to the same polymer matrix as studied here.

This strategy was limited by the divergence that has been previously described to happen with high λ values and high α values. The range of alpha values was therefore constrained in the algorithm so that the model would converge for any situation tested in the nanocomposite experiments with a σ value as low as 0.6. This was done in a stepwise fashion to fall in the convergence area as shown in Figure 8. This constraint could be limiting the accuracy of the polymer model predictions but is essential in order to model the nanocomposite.

Nanocomposite Model Fitting

The parameters that will be optimized to give more accurate predictions to nanocomposite viscosity are the polymer-particle interaction parameter and the particle-particle interaction parameter, σ and C_I respectively. Previous strategies to try to quantify C_I have used steady-state CNF orientation data because of the strong affect it has on particle orientation (Kagarise 2009). Another strategy was fitting the fourth order orientation tensor, a, to account for error in steady-state viscosity predictions as a function of shear rate and then again as a function of C_I. These algebraic correlations were then equated to give an expression for C_I as a function of shear rate based only on steady-state viscosity data (Kagarise 2009). C_I has not previously been fit to transient data. No previous works have attempted to quantify σ past the assumption of it being equal to one.

σ and C_I are assumed to be independent of CNF loading, shear rate and all other variables. The goal is to find the best global values for these parameters. These parameters cannot be physically measured and therefore must be treated as fitting parameters. The fitting space was created in a full factorial design of σ and C_I. A σ value equal to 1 has the mathematical meaning of no interactions between the nanoparticle and the polymer while a value of zero gives the most interaction possible through the model. This design space was limited to the range of σ values where the solution converged. This caused the range of σ values to be tested in this project to be limited to 0.6-1 with a spacing of 0.02. The magnitude of C_I is related to the intensity of interactions between nanoparticles and does not have physical limits on its value (Kagarise, 2009). Lower magnitudes result in more highly aligned particles in the direction of flow. Based on previous approximations for C_I, a range from 0.01-0.10 was chosen at

intervals of 0.002. In review, the model was tested at values of σ from 0.6-1.0 with an interval spacing of 0.02 and values of C_I from 0.01-0.10 with an interval spacing of 0.002 in a full factorial design to give a total of 966 possible combinations tested.

For each combination of parameter values, equations (1)-(4) were solved for all experimental conditions that were tested. These included, shear rates of 0.1 and 1 s^{-1}, forward and reverse flows with rest times of 0, 20 and 400 seconds and for both 2wt% and 5wt% CNF composites. For all simulations the CNF orientation was assumed to be initially completely randomly oriented. This assumption is probably not completely valid as some amount of orientation most likely developed when the sample was pressed into discs. The error was calculated in an identical fashion to the transient viscosity error calculation in the polymer fitting, shown in Equation (37), with the exception that there were more experiments to be taken into account. The result of these error calculations is shown in Figure 12. The overall minimum error was found to be at σ = 0.84 and C_I = 0.036. A MATLAB program was created for iteration and error calculation for both the forward and reverse flows. The errors from each were then divided by the number of experiments before being added together so equal weighting was given to both the forward and reverse directions. The MATLAB programs for calculating error in forward and reverse flows are shown in Appendix C and D respectively.

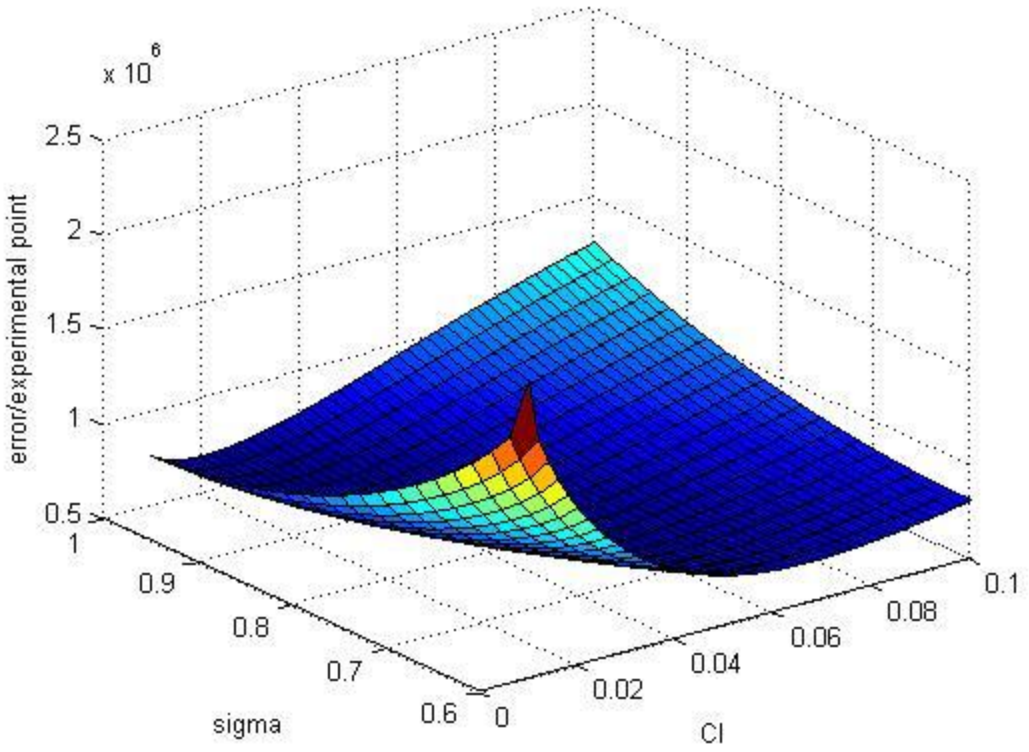


Figure 12: Error Surface for $\boldsymbol{\sigma}$ and $\boldsymbol{C_I}$ combinations

The optimal value of sigma is less than one and this suggests the model is accounting for some interactions between the CNFs and the polymer matrix. The changing of σ and C_I values mainly affected the later portion of the transient viscosity curve, closer to steady-state. Fitting these values especially helped the model be more accurate in making predictions in the stress overshoot and steady-state viscosity value. The stress overshoot is an increase in stress past that of the steady-state value before decaying back to the steady-state magnitude. The effect of the values of these two parameters can be seen in Figure 13 for both forward and reverse flows. The model fit to experimental data for every trial can be seen in Appendix E.

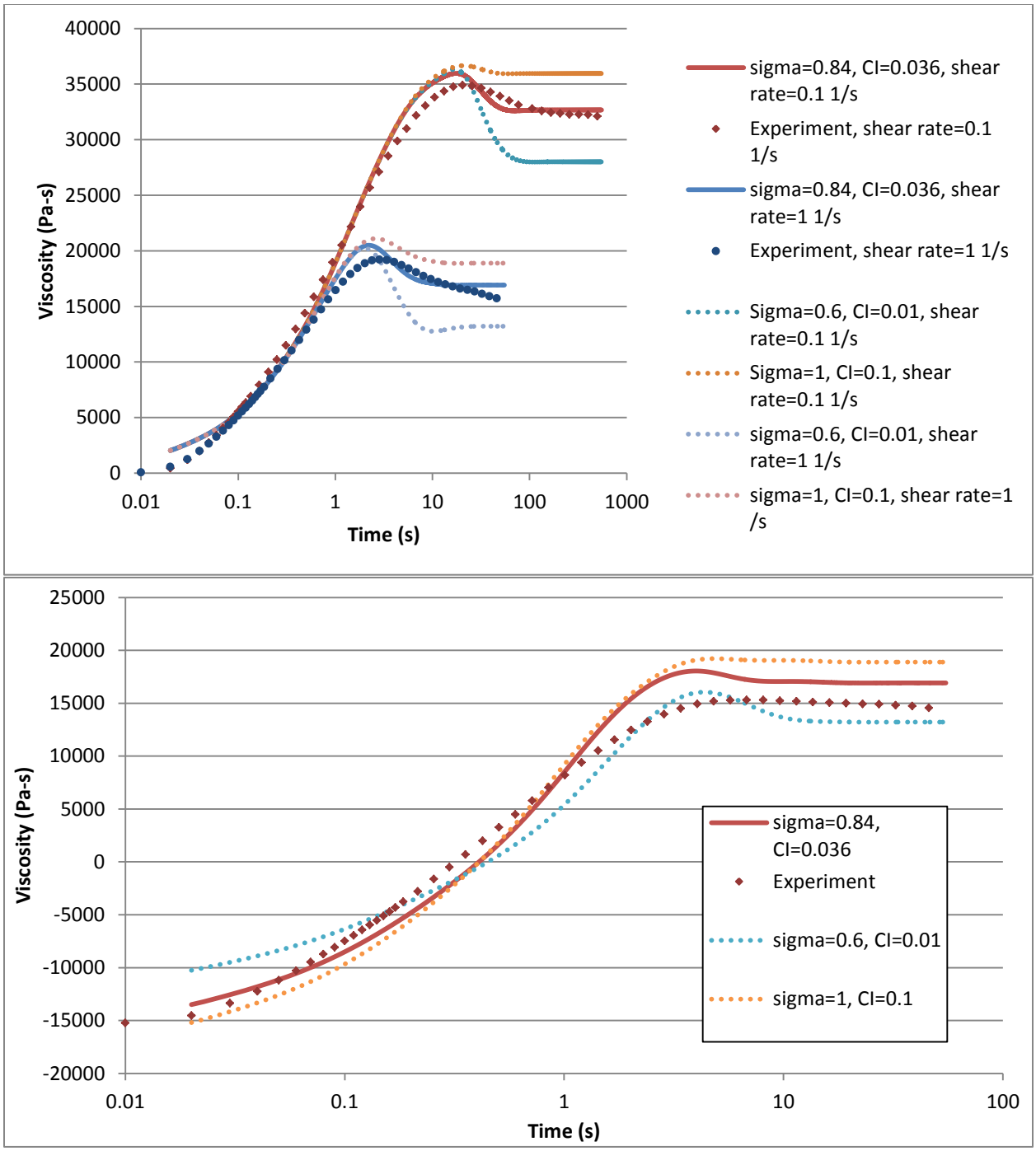


Figure 13: Effect of changing sigma and C_I values shown for forward flows at shear rates of 0.1 and 1 s^{-1} (Top) and a reverse flow at a shear rate of 1 s^{-1} after a 0 s rest (Bottom) both at 2wt% CNFs

The addition of CNFs to the polymer matrix is found both experimentally and by the model to give an increase in viscosity over the pure polymer. Figure 14 shows the difference in viscosity of a nanocomposite over the pure polymer for the model and experiments in forward

flows. The results from both experiments and the model predictions suggest viscosity increases with higher loading levels of CNFs. The area where the highest increase over the pure polymer is seen corresponds to the time when the sample is exhibiting a stress overshoot. This stress overshoot also appears to be affected by CNF loading with larger loadings giving larger stress overshoots.

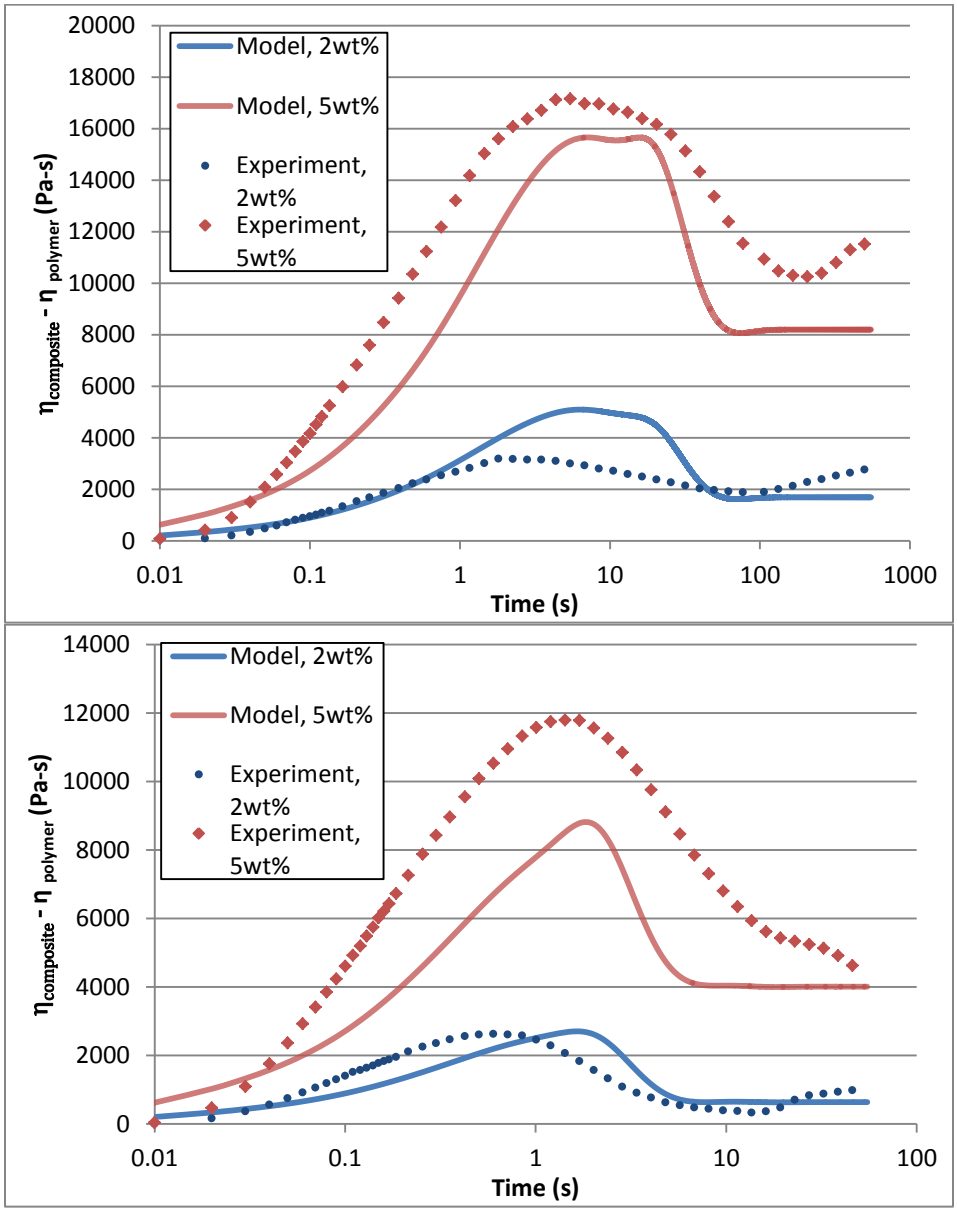


Figure 14: Additional viscosity of the nanocomposite over the pure polymer in shear flows at shear rates of 0.1 s^{-1} (Top) and 1 s^{-1} (Bottom).

The model is also capable of predicting the orientation of the nanoparticles at any point during a transient shear flow. As previously mentioned, the orientation of the nanoparticles significantly affects the resulting nanocomposite's mechanical, thermal and electrical properties (Miyazono *et al.*, 2011). This makes the nanoparticles orientation predictions important for predicting a product's final properties. Fitting to experimental orientation data was out of the scope of this project but it is worth noting the large affect that a change in C_I value has on the orientation evolution model predictions. This is illustrated in Figure 15. The range of C_I values tested in this study changes steady-state a_{11} predictions from 0.56 to 0.89. The a_{11} component of the second-order orientation tensor describes the orientation in the direction of flow. The previous work done fitting C_I to steady-state particles orientation data resulted in C_I=0.031 (Kagarise, 2009). The closeness in the result of this study suggests that the found optimal value of C_I will give accurate predictions for steady-state particle orientations as well as for transient viscosity.

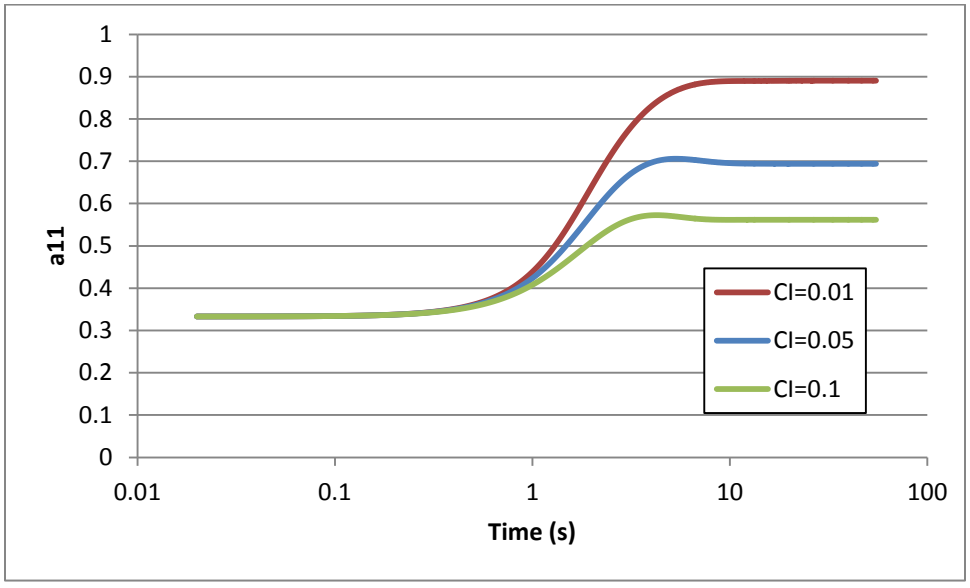


Figure 15: Change in the second-order orientation component a_{11} with change in C_I value for a 5wt% sample at a shear rate of 1 s^{-1}.

The orientation predictions can be used to offer an explanation for why increasing CNF loading increases the stress overshoot as shown previously in Figure 14. The component of the second-order orientation tensor, a_{11}, describes the amount of orientation in the direction of flow. Figure 16 shows the change in a_{11} alongside the viscosity during a 2wt% shear. It can be noted that the orientation in the direction of flow drastically changes during the time that the viscosity is exhibiting the stress overshoot. This suggests that the increase in stress overshoot from the addition of CNFs is caused by additional stress that is added from the CNFs reorienting in the sample from a random distribution to an aligned distribution in the direction of flow.

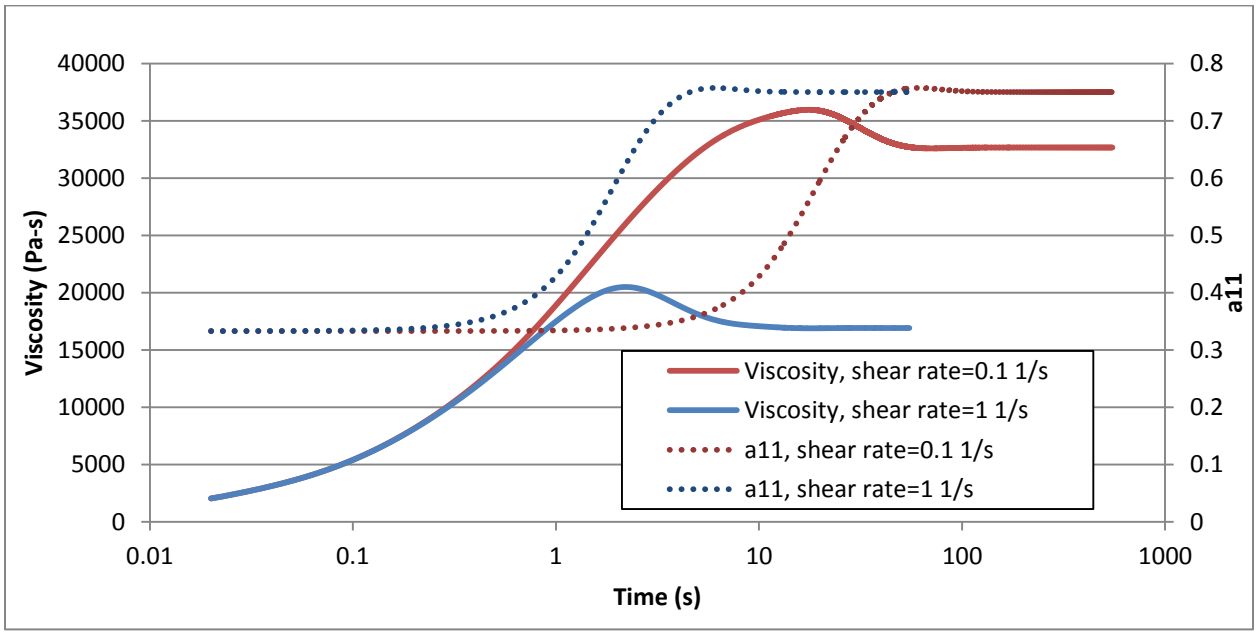
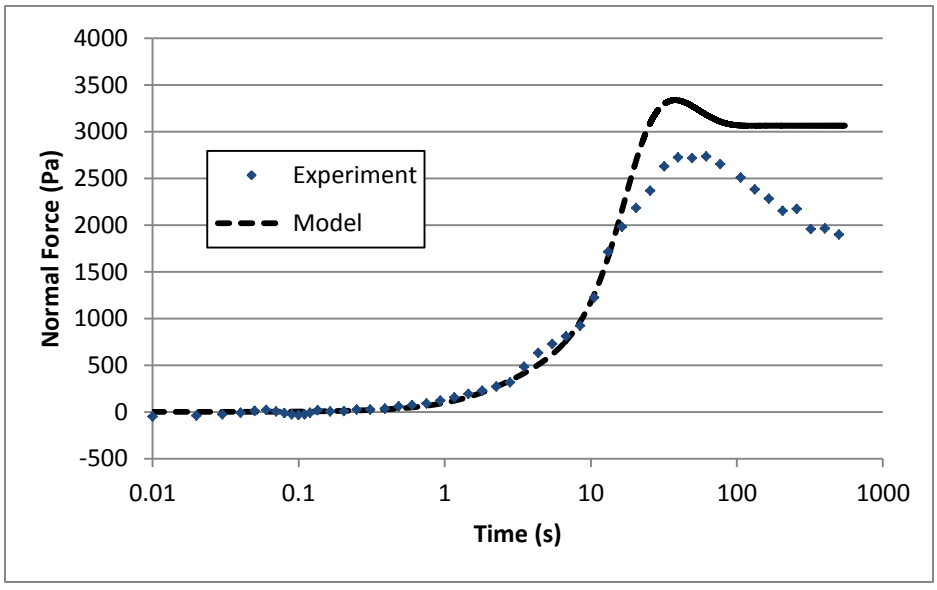


Figure 16: Correspondence of stress overshoot and change in orientation of CNFs

The model used in this study calculates all components of the stress tensor and is therefore not limited to just calculating viscosity. The normal force can also be calculated according to Equation (34).

Experimental data for normal stress was taken at the same time as viscosity data during testing. Normal stress was not used in the fitting of any parameters due to its relatively large amount of error in the experimental data when compared to viscosity experimental data. However, the experimental results can still be compared to model predictions in order to gain understanding of the accuracy of the model in predicting normal stress. Model predictions and experimental data for normal stress are shown in Figure 17 for a forward flow. The remaining trials of normal stress data and predictions can be found in Appendix F. The normal stress predictions show good qualitative agreement with experimental data in general. With a shear rate of 1 s^{-1} the predictions consistently give the rise in normal stress earlier than was found in experimental data. With a shear rate of 0.1 s^{-1}, the predictions give the increase at the proper time but over-predict the steady-state value of the normal stress. The normal force is also an important property during processing and having a model that can give accurate predictions for normal force as well as viscosity is important for understanding how the composite will behave during processing.



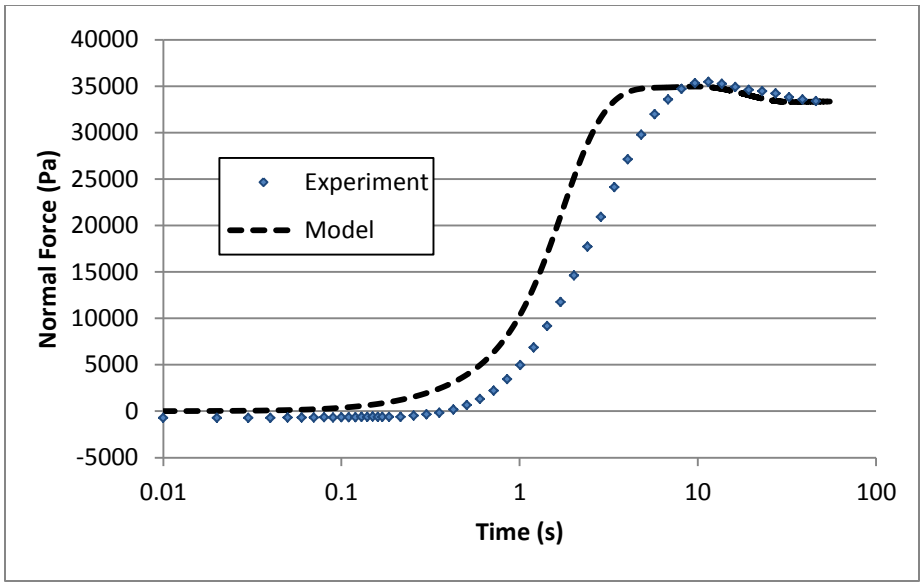


Figure 17: Normal force experimental data and model predictions in a forward flow at 2wt% CNFs at a shear rate of 0.1 s^{-1} (Top) and 1 s^{-1} (Bottom).

Small Amplitude Oscillatory Shear (SAOS) Modeling

The proposed model can also be validated against SAOS experiments. SAOS experiments are performed by shearing the sample in one direction until some specified strain is reached and then the shear direction is reversed until the specified strain is reached again and the process repeats giving strain that oscillates as a sine wave in the sample with respect to time with a frequency ω. The strain, γ, in these experiments can be expressed according to Equation (39) where γ^0 is the maximum strain achieved. The shear rate is defined as the derivative of the strain with respect to time. The resulting expression for the shear rate is shown in Equation (40). The stress wave resulting from SAOS experiments can be expressed as the sum of the in-phase and out-of-phase stress as given by Equation (41). G' and G" are then defined as the ratios of stress and strain amplitudes. The experimental values for G' and G" are defined as the values that give

the stress prediction from Equation (41) the closest fit to the experimentally produced stress wave in the SAOS experiments.

$$\gamma(t) = \gamma^0 \sin \omega t \tag{39}$$

$$\frac{d\gamma}{dt} = \dot{\gamma}(t) = \gamma^0 \omega \cos \omega t \tag{40}$$

$$\tau_{12}^{Gs} = G'(\omega)\gamma^0 \sin \omega t + G''(\omega)\gamma^0 \cos \omega t \tag{41}$$

The pure polymer model has an analytical solution for G' and G" from the Gieskus model as shown in Equation (35) and Equation (36), respectively. When the model is expanded to include nanoparticles, no such analytical solution is known. In order for G' and G" to be solved for in a nanocomposite model, the SAOS experiments are modeled to make predictions for the resulting stress wave in the sample. The values for G' and G" are then optimized by comparing the stress prediction from Equation (41) with the stress prediction from the model. A MATLAB program was created to model the SAOS experiments. This program is shown in Appendix G. This is the first time this model has been applied to SAOS experiments or any flows that have a strain rate that changes with time. The constant used for shear rate previously in the model must now be replaced by Equation (40). This affects Equations (2), (3) and (4) of the model and all were changed accordingly. The stress prediction from the model and from Equation (41) using experimentally determined G' and G" is shown in Figure 18 at an intermediate frequency of 1 radian/s in a 2wt% sample.

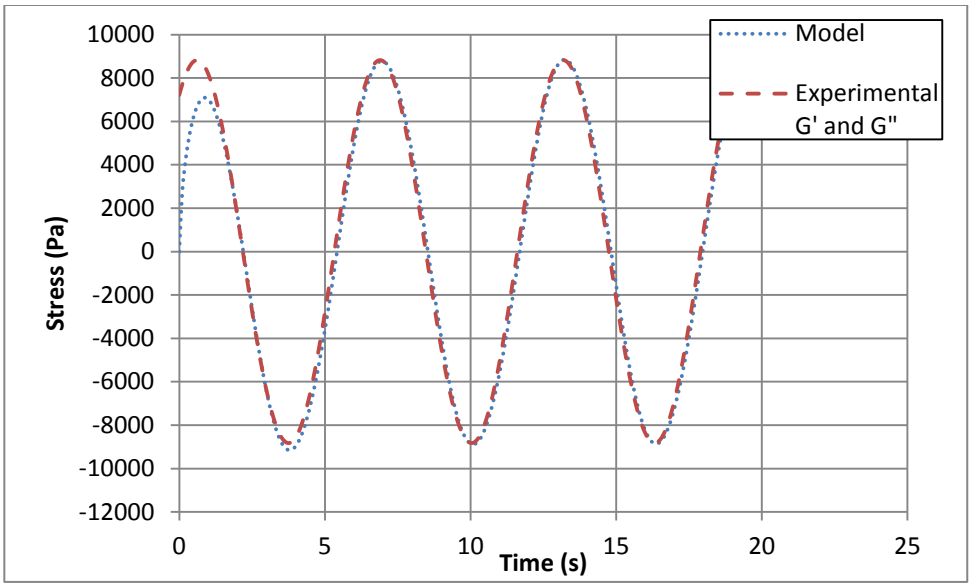


Figure 18: Stress prediction for SAOS flow in a 2wt% sample from model prediction and from Equation (41) using experimental G' and G" values.

A strategy was subsequently developed for finding the G' and G" values that give the best fit to the model prediction of stress. This was done by first solving the model to get a prediction for the stress wave. An initial guess for G' and G" was then provided and Equation (41) solved. The error between these two predictions for the stress waves was calculated according to Equation (42) for a total time equal to 3 periods of the sine waves. As can be seen in Figure 18, the model prediction has some initial period where the stress has not reached a steady oscillation. For this reason, the error between the two stress waves was not calculated until one complete period of the sine wave had passed. This procedure was then iterated according to the "active-set" algorithm in MATLAB until a minimum in error was reached. This process was repeated for every frequency that was experimentally tested. The MATLAB program used for this optimization can be found in Appendix H.

$$error = \sum_{j=1}^{113}[\tau_{12}^{c}(t_j) - \tau_{12}^{Gs}(t_j)]^2 \quad (42)$$

This strategy for solving for G' and G" values from the model can be validated by running it for the pure polymer case where it can be compared to the analytical solution results

calculated from Equations (35) and (36). The results from both methods are shown in Figure 19. There is very close agreement between the two methods. The analytical Gieskus solution is a proven method and the close agreement of this new method suggests this new method is working as desired and will be able to make reliable predictions for nanocomposite systems.

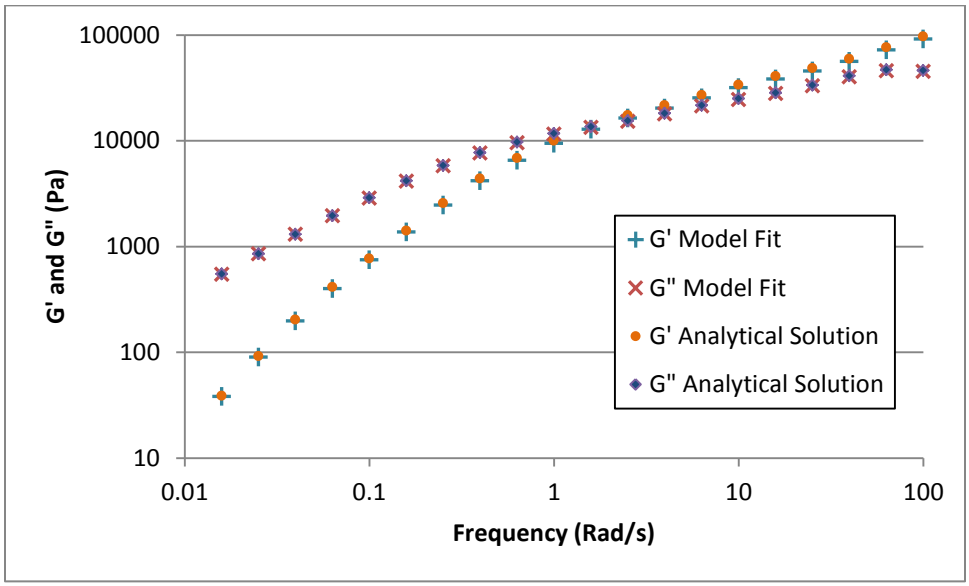
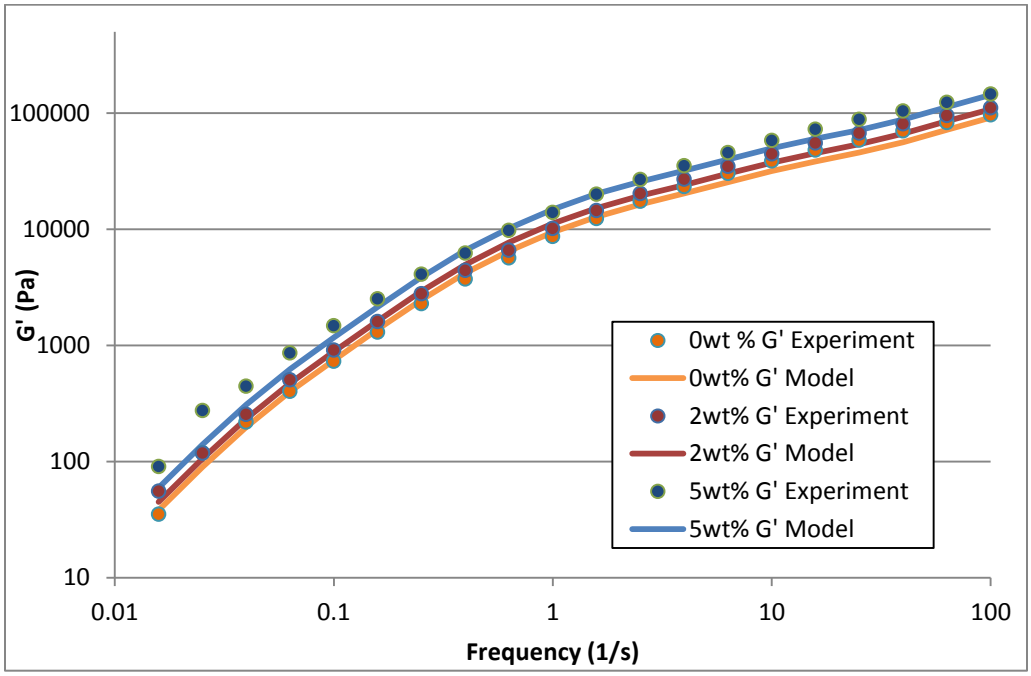


Figure 19: Comparison of G' and G" predictions from fitting to model stress wave and from Gieskus analytical solution for the pure polymer.

This program was run to for 0wt%, 2wt% and 5wt% CNF loadings. The resulting values of G' and G" were then compared to experimental values as shown in Figure 20. The predictions for both G' and G" are fairly accurate at low to intermediate frequencies with the exception of the 5wt% G' predictions which are considerably lower than experimental data. Predictions are also low from a frequency of approximately 4-40 radians/s for G' and 1.5-25 radians/s for G". This error is seen both for the pure polymer and the composite suggesting that this error comes from the fitting of the pure polymer parameters rather than the composite models inability to predict these type of flows. Predictions at the extreme high end of frequencies tested again show reasonable accuracy when compared to experimental data. It should be noted that G' and G"

predictions would be more accurate if η_p and λ were fit only using G' and G" data along with the analytical solution given by the Gieskus model. However as stated previously, only part of the weighting in determining η_p and λ was given by this method and the other part was given by fitting to transient shear viscosity experiments for the pure polymer.



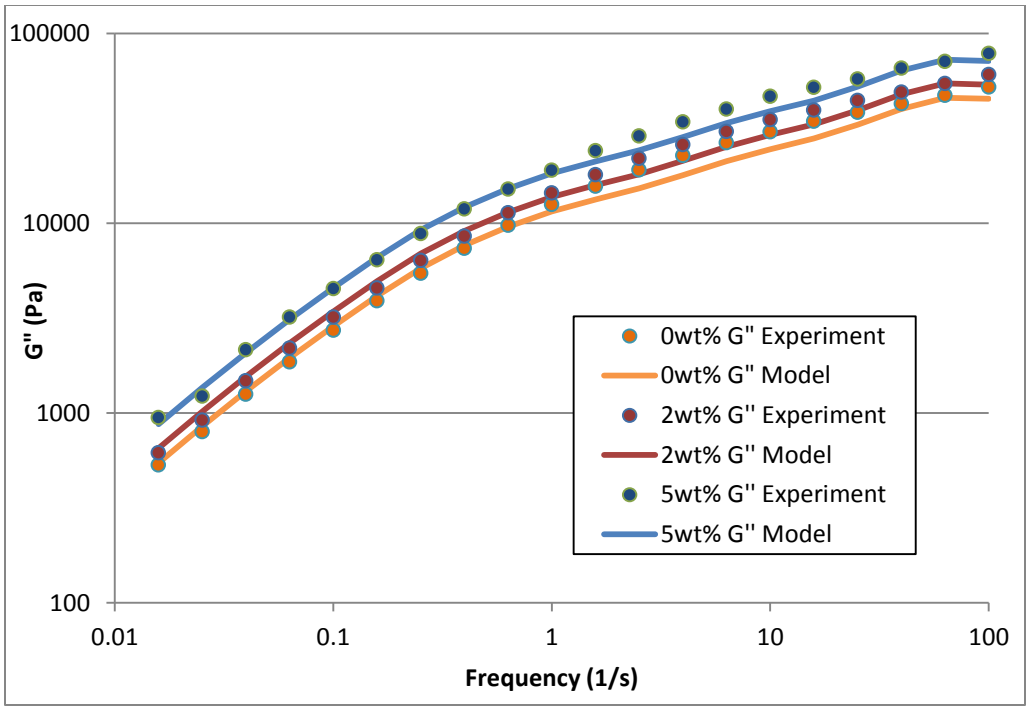


Figure 20: G' (Top) and G" (Bottom) predictions from the model and experimental data

The model appears to predict a shift, that is fairly independent of frequency, to higher G' and G" values with increasing CNF loading on a logarithmic scale. This was investigated by calculating the ratio of increase the logarithm of G' and G" for a nanocomposite over the logarithm of the pure polymer values as seen in Equation 43 where X is the wt% CNFs in the nanocomposite.

$$Ratio\ of\ Increase\ on\ a\ Logarithmic\ Scale = \frac{\log G'_{Xwt\%}}{\log G'_{0wt\%}} \tag{43}$$

If the increase in G' and G" values with increasing CNF loading is truly a simple upward shift from multiplying by some factor than this ratio will be the same for all frequencies. The average ratios and standard deviations are shown in Table 3.

Table 3: Ratio of increase of the log(G) from 0wt% to both 2 and 5wt% calculated according to Equation 43. Results are averaged over all frequencies tested.

CNF wt%	2				5			
Data Source	Model Prediction		Experimental		Model Prediction		Experimental	
	G'	G"	G'	G"	G'	G"	G'	G"
Average Ratio from all Frequencies	1.189406	1.190504	1.187701	1.158248	1.575318	1.587844	1.748875	1.574915
Standard Deviation	0.003438	0.004116	0.107577	0.010511	0.004665	0.005692	0.317904	0.086355

The ratio increases with increasing CNF loading as expected for both the model predictions and the experimental data for both G' and G". The model predictions give a very low standard deviation in ratio with different frequencies supporting the idea that the increase comes from a simple multiplying factor valid for all frequencies. This factor can be used to predict the increase in G' or G" from the pure polymer according to Equation 44 where F is the factor as determined by the average result of Equation 43 for all frequencies as shown in Table 3.

$$\log G_{Xwt\%} = F \times \log G_{0wt\%} \tag{44}$$

Interestingly, the calculated factor is roughly the same for G' and G". This can be seen graphically in Figure 21. The factors also seem to increase linearly with increasing CNF loading. The R^2 value for a linear trend line fit to the G' and G" model predictions ratios are 0.994 and 0.993, respectively. For the experimentally determined ratios the R^2 values are 0.973 and 0.981 for G' and G", respectively. This information can be used to make rough predictions of G' and G" for nanocomposites with a very limited amount of data. If the assumptions of the multiplying factor being the same for every frequency, the same for both G' and G" and that the multiplying factor increases linearly with CNF loading, only the G' and G" data from the pure polymer and

one single data point of G' or G" from a nanocomposite of any loading would be needed to make estimations of G' and G" at any loading and frequency. These assumptions are all supported by the above findings in the range that was tested in this study.

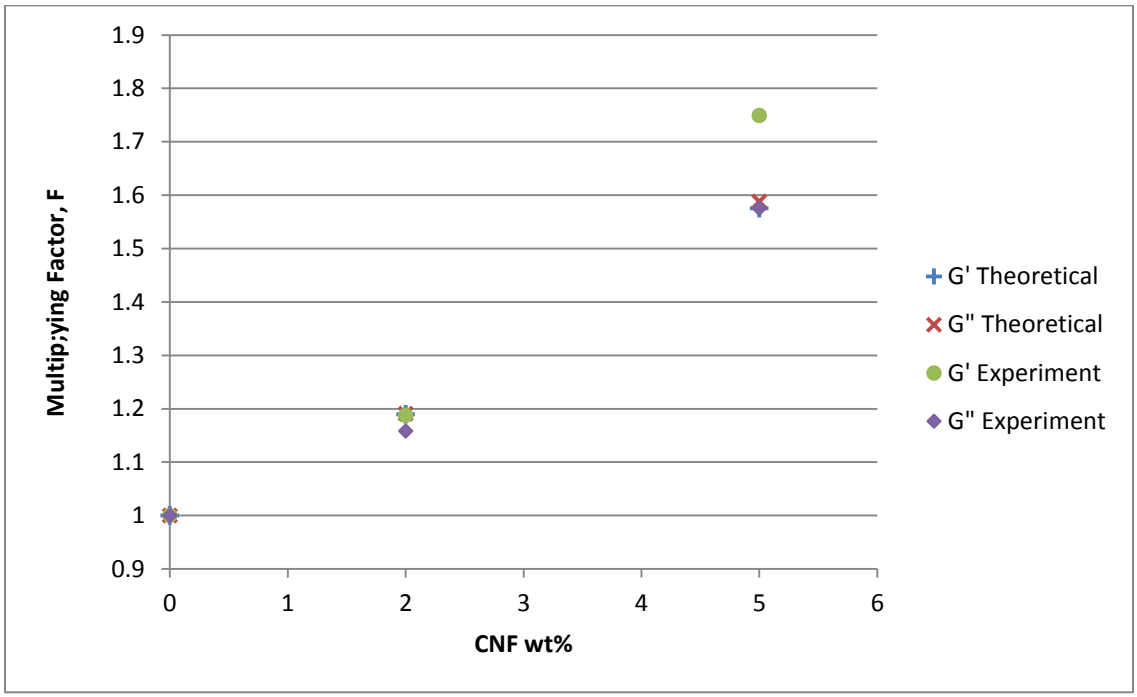
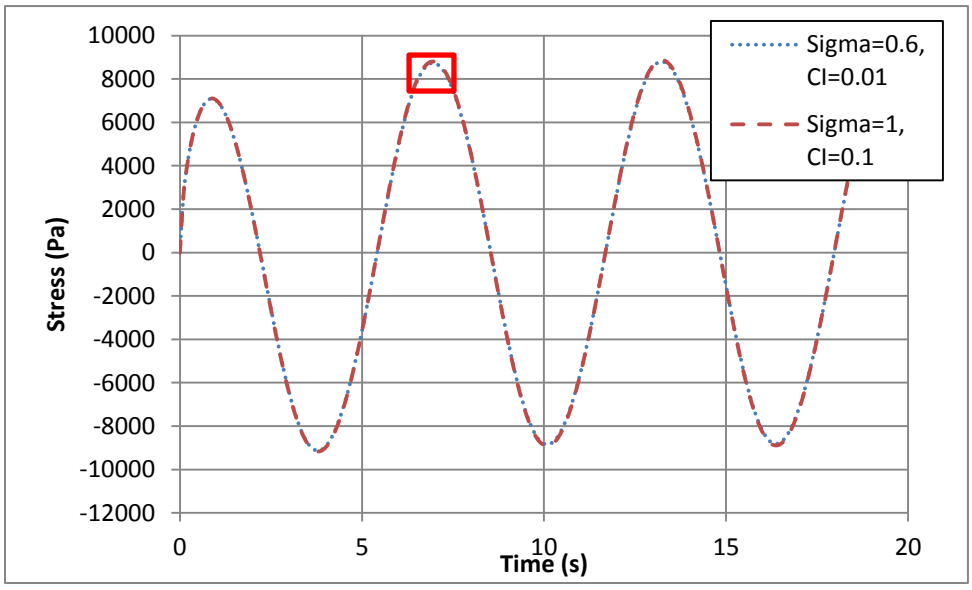


Figure 21: Multiplying factor as a function of CNF loading

An attempt was made to investigate whether the optimized values for σ and C_I were also good fits for SAOS flows. The initial strategy was to do a fitting of σ and C_I to find the optimal values that would give G' and G" model predictions that most closely matched experimental G' and G" data. The results showed that SAOS flows are very weak functions of σ and C_I and fitting them to experimental G' and G" data made little to no impact. Figure 22 shows stress wave predictions from two combination of σ and C_I that are located at opposite corners of the design space for these variables. These are the same combinations shown earlier that produced significantly different predictions for regular shear flows as seen in Figure 13. In order to make

out a difference between the two predictions, it is necessary to zoom in on an individual peek. This result is not altogether unexpected after inspecting model predictions of normal forward shear flows as seen again in Figure 13. The effect of σ and C_I values is not readily apparent until a relatively large amount of strain is generated, somewhere around a strain of one. The SAOS flows never generate a large amount of strain and therefore do not show much variation with changing values of σ and C_I. If large amplitude oscillatory shear experiments were performed they might provide a basis for fitting σ and C_I with these relatively simple experiments because of the higher strains achieved.



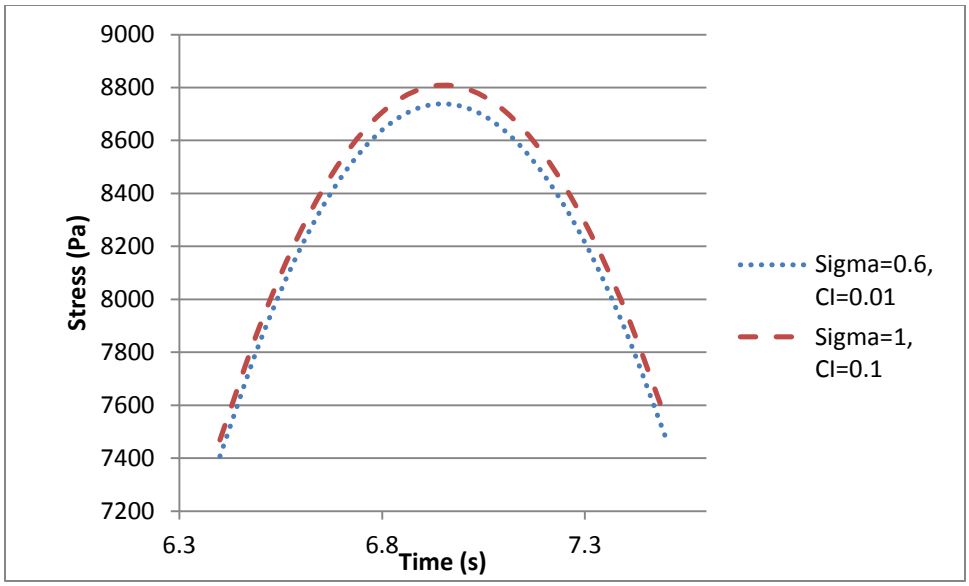


Figure 22: SAOS stress wave model predictions at different combinations of $\boldsymbol{\sigma}$ and $\boldsymbol{C_I}$ for entire wave (Top) and zoomed on a peak (Bottom) as denoted by the red box.

CONCLUSIONS AND FUTURE WORK

This study presented a previously developed model for predicting nanocomposite stress and orientation. The goal was to optimize several parameters in this model and apply the model to new flow regimes and evaluate model accuracy.

The values for the particle-polymer interaction parameter, σ, and the particle-particle interaction parameter, C_I, were optimized for transient shear viscosity predictions for both the forward and reverse directions. No previous study has attempted to fit the value of σ further than the assumption of it being equal to one which gives no interactions between the nanoparticle and the polymer. Only limited work has been previously done to fit a value to C_I and this is the first

study that includes a fitting to transient data. The optimum values for this particular melt-blended nanocomposite of polystyrene and CNFs were found to be $\sigma = 0.84$ and $C_I = 0.036$. With these optimized parameters, the model is able to make more accurate predictions of shear flows. These parameters mainly affect the magnitude and shape of the stress overshoot and the subsequent steady-state viscosity value and therefore the model is particularly more accurate in this region.

The purpose of these parameters in the model is to account for interactions that are specific to both the nanoparticle and the polymer. The optimum values found in this study may not be the best choice for other systems but the procedure developed in this study can be used to find optimum values for these parameters in other systems.

The model with optimized parameters was also evaluated for its ability to predict the normal force during shear flows. The predictions gave qualitatively correct shape of the transient curves except for the rise in normal force is predicted slightly early in flows with a shear rate of 1 s^{-1}. Reasonable agreement in magnitude is shown for all trials. Overall, the proposed model can be a useful tool in predicting the normal force in addition to the transient shear viscosity.

The model was also applied to SAOS flows for the first time. The stress wave produced from this type of flow is modeled based on the parameters found in this study. The values of σ and C_I were found to have very little effect on model predictions. The simulated stress waves were used to find the G' and G" values that best fit model predictions. G' and G" values increase with increasing CNF loading. An estimate of this increase can be obtained from multiplying a simple factor to the log of the G' or G" value. The values for G' and G" are then compared to experimental values to assess the model's accuracy in predicting stress in these types of flows. The model shows good agreement to experimental data except in a range of frequencies approximately from 1.5-16 radians/s. The error in this region is consistent for the pure polymer

as well as the nanocomposite suggesting that the error in this region is based on the fitting done to pure polymer parameters instead of the model itself.

Future work in this area will include applying the procedure for optimizing σ and C_I from this study to extensional flows and evaluating whether the values found in this study are the best values for extensional flows. Another option for future work would be using the procedure laid out by Kagarise to experimentally measure the orientation of the nanofibers at various points during a flow (Kagarise, 2009). This data could then be used to fit C_I to give the best predictions for orientation and compare to the value found in this study.

The SAOS experiments did not provide a good basis for doing fitting of σ and C_I because the stress at the low strains achieved is only a weak function of these parameters. If large amplitude oscillatory shear experiments were performed they may provide a simple basis for fitting these parameters and comparing to the optimum values determined in this study. The SAOS experiments could be used for other fittings than to σ and C_I. The aspect ratio of the CNFs also affects the stress. This is a labor intensive process to measure the aspect ratio experimentally and requires an electron microscope. A fitting to SAOS experiments performed on CNF nanocomposites with an unknown aspect ratio could be used approximate an average value for the aspect ratio.

REFERENCES

Advani, S.G. and Tucker, C.L. III. "The use of tensors to describe and predict fiber

orientation in short fiber composites." J. Rheol., 31, (1987), 751-784.

Advani, S.G. and Tucker, C.L. III. "Closure approximations for three-dimensional structure tensors" J. Rheol. 34, (1990), 367-386.

Azaiez, J. "Constitutive equations for fiber suspensions in viscoelastic media." J. Non-Newtonian Fluid Mech., 66, (1996): 35-54.

AZoNano. "Graphitic Carbon Nanofibers - Properties and Applications of Pyrograf III." *Graphitic Carbon Nanofibers - Properties and Applications of Pyrograf III*, (2012).

Biercuk, M.J.; Llaguno, M.C.; Radosavljevic, M.; Hyun, J.K.; Johnson, A.T.; and Fischer, J.E. "Carbon nanotube composites for thermal management." Appl. Phys. Lett., 80, (2002): 2767-2769.

Bird, R.B.; Curtiss, C.F.; Armstrong, R.C.; and Hassager, O. *Dynamics of Polymeric Liquids.* Wiley, New York, (1987).

C.L. Lake "Carbon Nanofiber Composites: From Innovative R&D to Commercial Reality"

Caldiera, G.; Maia, J.M.; Carneiro, O.S.; Covas, J.A.; and Bernardo, C.A. "Production and characterization of innovative carbon fiber-polycarbonate composites." Polym. Compos., 19, (1998): 147-151.

Caldiera, G.; Maia, J.M.; Carneiro, O.S.; Covas, J.A.; and Bernardo, C.A. "Production and characterization of innovative carbon fiber-polycarbonate composites." Polym. Compos., 19, (1998): 147-151.

Carneiro, O. S. and Maia, J.M. "Rheological behavior of short carbon fiber/thermoplastic composites. Part I: The influence of fiber type, processing conditions and level of incorporation," Polym. Compos., 21, (2000): 960–969.

Carneiro, O. S.; Covas, J.A.; Bernardo, C.A.; Caldeira, G.; Van Hattum, F.W.J.; Ting, J.-M.; Alig, R.L.; Lake, M.L. "Production and assessment of polycarbonate composites reinforced with vapour-grown carbon fibres," Compos. Sci. Technol., 58, (1998): 401–407.

Ceccia, S.; Ferri, D.; Tabuani, D.; and Maffettone, P.L. "Rheology of carbon nanofiber-reinforced polypropylene." Rheol. Acta, 47, (2008): 425-433.

Choi, E.S.; Brooks, J.S.; Eaton, D.L.; Al-Haik, M.S.; Hussaini, M.Y.; Garmestani, H.; Li, D.; and Dahmen, K. "Enhancement of thermal and electrical properties of carbon nanotube polymer composites by magnetic field processing." J. Appl. Phys., 94, (2003): 6034-6039.

Cortes P.; Lozano K.; Barrera E.V.; Bonilla-Rios J. "Effects of nanofiber treatments on the properties of vapor-grown carbon fiber reinforced polymer composites." J Appl. Polym. Sci., 89, (2003): 2527–34.

Du, F.; Guthy, C.; Kashiwagi, T.; Fischer, J.E.; Winey, K.I. "An infiltration method for preparing single-wall nanotube/epoxy composites with improved thermal conductivity." J. Polym. Sci., Part B: Polym. Phsy., 44, (2006): 1513-1519.

Dyke, C.A. and Tour, J.M. "Covalent functionalization of single-walled carbon nanotubes for materials applications." J. Phys. Chem. A, 108, (2004): 11151-11159.

Eder, E. Hammel, T. Schmitt, X. Tang, M. Trampert, K. Mauthner, Electrovac AG "Carbon Nanofiber Composites a Commercial nano-application"

F. Folgar and C.L. Tucker (1984) Orientation behaviour of fibres in concentrated suspensions. J Reinf Plast Compos 3: 98-119.

Geng, H.; Rosen, R.; Zheng, B.; Shimoda, H.; Fleming, L.; Liu, J.; and Zhou, O. "Fabrication and properties of composites of poly(ethylene oxide) and functionalized carbon nanotubes." Adv. Mater., 14, (2002): 1387-1390.

Jaing, X.; Bin, Y.; and Matsuo, M. "Electrical and mechanical properties of polyimide-carbon nanotubes composites fabricated by in situ polymerization." Polymer, 46, (2005): 7418-7424.

Kagarise, C.; Koelling, K.W.; Wang, Y.; and Bechtel, S.E. "A unified model for polystyrene-nanorod and polystyrene-nanoplatelet melt composites." Rheol. Acta, 47, (2008): 1061-1076.

Kagarise, C.; Miyazono, K.; Mahboob, M.; Koelling, K.W.; and Bechtel, S.E. "A constitutive model for characterization of shear and extensional rheology and flow induced orientation of carbon nanofiber/polystyrene melt composites." J. Rheol., 55(4), (2011): 781-807.

Kagarise, C.; Xu, J.; Wang, Y.; Mahboob, M.; Koelling, K.W.; and Bechtel, S.E. "Transient shear rheology of carbon nanofiber/polystyrene melt composites." J. Non-Newtonian Fluid Mech., 165, (2010): 98–109.

Kagarise, C.D. "Rheological characterization and modeling of micro- and nano-scale particle suspensions." Dissertation, The Ohio State University, (2009).

Lozano, K.; Bonilla-Rios, J.; and Barrera, E.V. "A study on nanofiber-reinforced thermoplastic composites (II): Investigation of the mixing rheology and conduction properties." J. Appl. Polym. Sci., 80, (2001): 1162–1172.

Lozano, K.; Yang, S.; and Zeng, Q. "Rheological analysis of vapor-grown carbon nanofiber-reinforced polyethylene composites." J. Appl. Polym. Sci., 93, (2004): 155-162.

Ma, H.; Zeng, J.; Realff, M.L.; Kumar, S.; and Schiraldi, D.A. "Processing, structure, and properties of fibers from polyester/carbon nanofiber composites." Compos. Sci. Technol., 63, (2003): 1617-1628.

Meincke, O.; Kaempfer, D.; Weickmann, H.; Friedrich, C.; Vathauer, M.; and Warth, H. "Mechanical and electrical conductivity of carbon-nanotube filled polyamide-6 and its blends with acrylonitrile/butadiene/styrene." Polymer, 45, (2004): 739-748.

Miyazono, K.; Kagarise, C.D.; Koelling, K.W.; Mahboob, M.; and Bechtel, S.E. "Shear and extensional rheology and flow-induced orientation of carbon nanofiber/polystyrene melt composites." Journ. Of Appl. Polym. Sci., 119, (2011): 1940-1951.

Modi S.H.; Dikovics K.B.; Gevgilili H; Mago G; Bartolucci S.F.; Fisher F.T.; Kalyon D.M. "Nanocomposites of poly(ether ether ketone) with carbon nanofibers: Effects of dispersion and thermo-oxidative degradation on development of linear viscoelasticity and crystallinity." Polymer, 51, (2010): 5236-5244.

Ramasubramaniam, R.; Chen, J.; and Liu, H. "Homogeneous carbon nanotube/polymer composites for electrical applications." Appl. Phys. Lett., 83, (2003): 739-748.

Tucker, C.L. III. "Flow regimes for fiber suspensions in narrow gaps." J. Non-Newtonian Fluid Mech., 39, (1991): 239-268.

Varela-Rizo H.; Bittolo-Bon S.; Rodriguez-Pastor I.; Valentini L.; Martin-Gullon I. "Processing and functionalization effect in CNF/PMMA nanocomposites." Composites Part A: Applied Science and Manufacturing, 43, (April 2012): 711-721.

Varela-Rizo H.; Montes de Oca G.; Rodriguez-Pastor I.; Monti M.; Terenzi A.; Martin-Gullon I.; "Analysis of the electrical and rheological behavior of different processed CNF/PMMA nanocomposites." Composites Science and Technology, 72, (January 2012): 218-224.

W. Murch "The Effect of Flow-induced Orientation on the Rheology of Carbon Nanofibers/Polystyrene Composites During Flow Reversal Experiments" The Ohio State University Knowledge Bank, (2011).

Wang, Y.; Xu, J.; Bechtel, S.E.; and Koelling, K.W. "Melt shear rheology of carbon nanofiber/polystyrene composites," Rheol. Acta, 45, (2006): 919–941.

Xu, J.; Chatterjee, S.; Koelling, K.W.; Wang, Y.; and Bechtel, S.E. "Shear and extensional rheology of carbon nanofiber suspensions." Rheol. Acta, 44, (2005): 537-562.

Yang, S.; Lozano, K.; Lomeli, A.; Foltz, H.D.; and Jones, R. "Electromagnetic interference shielding effectiveness of carbon nanofiber/LCP composites." Composites, Part A, 36, (2005): 691-697.

APPENDICES

Appendix A: MATLAB program capable of solving model equations and comparing to experimental data for transient shear forward and reverse flows

```
%Written By Monon Mahboob and Michael Smith
%Edited by William Murch Tim Kremer
%Modeling program for Transient Stress on NanoFiber Compounds


%This program makes model predictions for forward and reverse flows and
%compares them to Will Murch's experimental data
function [model experiment] = GeneralShear(r1,massfrac,sigma,CI,resttime)

%Original shear program by Monon Mahboob and Michael Smith
%Edited by William Murch and Tim Kremer
%Modeling program for Transient Stress on NanoFiber Compounds


%This program makes model predictions for forward and reverse flows and
%compares them to Will Murch's experimental data

if nargin==0
   r1=1;          % shear rate (initially positive)
   massfrac=0.05;  % mass fraction of CNFs
   resttime=0;     % rest time inbetween forward and reverse flows
   CI=0.036;       %particle-particle interaction parameter
   sigma=0.84;     %particle-polymer interaction parameter
end

re=33.0; %aspect ratio

chi=1.0*(re^2-1)/(re^2+1); %chi, function of aspect ratio
rf=1750.0; %fiber density
rs=1000.0; %polymer density


etap=[1100.046682   2920.193962 9277.693362 14316.47647 7408.914859]; %polymer viscosity term in Equation
(2)
lambda=[0.012287706 0.11728325  0.743829501 2.689760868 15.39198862]; %Relaxation time from Equation (2)
alpha=[0.50413207   0.678743122 0.680016347 0.317377081 0.27975612]; %Mobility factor from Equation (2)


r=r1;   %active shear rate (1/s)
modes=5;
shear_time=abs(55/r);   %length of simulation
tspan=0:0.01:shear_time;    %timespan with stepsized sent to ode solver
```

```
phi=1.0*rs*massfrac/(rf+(rs-rf)*massfrac);  %volume fraction
Ap=1.0*re^2/(3*log(sqrt(1.0*pi/phi)));  %A2, shape factor

orientation=[0.333, 0.0, 0.333];    %initial orientation assume random

tau_finalf=zeros(4,modes);

for x=1:modes
   initial=[0 0 0 0 orientation];  %initial conditions for ode solver
   [time1, yo]=ode23tb(@modepolymersub,tspan,initial);     %ode solver for solving Equations (2) & (4)
simultaneously
   export = [' mode ' num2str(x) ' forward calculation done'];
   disp(export)

   L=length(time1);

   if x==1
      tau1=zeros(L,modes);
      tau12=zeros(L,modes);
      tau2=zeros(L,modes);
      tau3=zeros(L,modes);
   end

   tau1(:,x)=yo(:,1);
   tau12(:,x)=yo(:,2);
   tau2(:,x)=yo(:,3);
   tau3(:,x)=yo(:,4);


   tau_finalf(:,x) = [tau1(L,x) tau12(L,x) tau2(L,x) tau3(L,x)]';

end

total11f=sum(tau1,2);
total12f=sum(tau12,2);
total22f=sum(tau2,2);
total33f=sum(tau3,2);

a11f=yo(:,5);
a12f=yo(:,6);
a22f=yo(:,7);
a33f=1-a11f-a22f;

disp('final values for T11p T12p T22p T33p Forward')
T_final= [total11f(L) total12f(L) total22f(L) total33f(L)];
disp(T_final)

disp('mode forward stress finals T11 T12 T22')
disp(tau_finalf)

disp('forward final values for a11 a12 a22 a33 =')
```

```
a_final = [a11f(L) a12f(L) a22f(L)]';
disp(a_final)

etaf=total12f./r;
coef=2.0*etaf*phi;

%%%%This section computes the fiber stress from Equation (3)
if r~=0
   disp('r~=0')
   tf12f=(coef*Ap*r).*((27.0.*(a11f.*a22f.*(1-a11f-a22f)-(a12f.^2).*(1-a11f-a22f))).*(-
1.0/35.0+1.0/7.0.*(a11f+a22f))+(a12f.^2).*(1-27.*(a11f.*a22f.*(1-a11f-a22f))+27.0.*(a12f.^2).*(1-a11f-a22f)));
   tf11f=(coef*Ap*r).*((27.0.*(a11f.*a22f.*(1-a11f-a22f)-(a12f.^2).*(1-a11f-
a22f))).*(3.0/7.0.*(a12f))+(a11f.*a12f).*(1-27*(a11f.*a22f.*(1-a11f-a22f))+27.0.*(a12f.^2).*(1-a11f-a22f)));
   tf22f=(coef*Ap*r).*((27.0.*(a11f.*a22f.*(1-a11f-a22f)-(a12f.^2).*(1-a11f-
a22f))).*(3.0/7.0.*(a12f))+(a22f.*a12f).*(1-27*(a11f.*a22f.*(1-a11f-a22f))+27.0.*(a12f.^2).*(1-a11f-a22f)));
   tf33f=(coef*Ap*r).*((27.0.*(a11f.*a22f.*(1-a11f-a22f)-(a12f.^2).*(1-a11f-a22f))).*(1.0/7.0.*(a12f))+((1-a11f-
a22f).*a12f).*(1-27.*(a11f.*a22f.*(1-a11f-a22f))+27.0.*(a12f.^2).*(1-a11f-a22f)));
else
   disp('r==0')
   tf12f=(2.0*phi.*total12f.*Ap).*((27.0*(a11f.*a22f.*(1-a11f-a22f)-(a12f.^2).*(1-a11f-a22f))).*(-
1.0/35.0+1.0/7.0*(a11f+a22f))+(a12f.^2).*(1-27*(a11f.*a22f.*(1-a11f-a22f))+27.0*(a12f.^2).*(1-a11f-a22f)));
   tf11f=(2.0*phi.*total12f.*Ap).*((27.0*(a11f.*a22f.*(1-a11f-a22f)-(a12f.^2).*(1-a11f-
a22f))).*(3.0/7.0*(a12f))+(a11f.*a12f).*(1-27*(a11f.*a22f.*(1-a11f-a22f))+27.0*(a12f.^2).*(1-a11f-a22f)));
   tf22f=(2.0*phi.*total12f.*Ap).*((27.0*(a11f.*a22f.*(1-a11f-a22f)-(a12f.^2).*(1-a11f-
a22f))).*(3.0/7.0*(a12f))+(a22f.*a12f).*(1-27*(a11f.*a22f.*(1-a11f-a22f))+27.0*(a12f.^2).*(1-a11f-a22f)));
   tf33f=(2.0*phi.*total12f.*Ap).*((27.0*(a11f.*a22f.*(1-a11f-a22f)-(a12f.^2).*(1-a11f-
a22f))).*(1.0/7.0*(a12f))+((1-a11f-a22f).*a12f).*(1-27*(a11f.*a22f.*(1-a11f-a22f))+27.0*(a12f.^2).*(1-a11f-
a22f)));
end

tauc12f=tf12f+total12f;
tauc11f=tf11f+total11f;
tauc22f=tf22f+total22f;
tauc33f=tf33f+total33f;
etacf=tauc12f./r;

%-------------------------------------------------------------------------
%   RELAXATION PERIOD
%-------------------------------------------------------------------------
if resttime~=0
tau_finalm=zeros(4,modes);
r=0;

for x=1:modes
   initial2=[tau_finalf(:,x); a_final];    %initial conditions for relaxation period set as final conditions from forward
flow
   tspan2=time1(end):0.01:resttime+time1(end); %tspan for rest period
   [time2, yo]=ode23tb(@modepolymersub,tspan2,initial2);   %ode solver for rest time
   export = [' mode ' num2str(x) ' relaxation calculation done'];
   disp(export)

   L=length(time2);

   if x==1
```

```
        tau1=zeros(L,modes);
        tau12=zeros(L,modes);
        tau2=zeros(L,modes);
        tau3=zeros(L,modes);
    end

    tau1(:,x)=yo(:,1);
    tau12(:,x)=yo(:,2);
    tau2(:,x)=yo(:,3);
    tau3(:,x)=yo(:,4);

    tau_finalm(:,x) = [tau1(L,x) tau12(L,x) tau2(L,x) tau3(L,x)]';

end

total11m=sum(tau1,2);
total12m=sum(tau12,2);
total22m=sum(tau2,2);
total33m=sum(tau3,2);

a11m=yo(:,5);
a12m=yo(:,6);
a22m=yo(:,7);
a33m=1-a11m-a22m;

disp('final relaxation values for T11p T12p T22p T33p')
T_final= [total11m(L) total12m(L) total22m(L) total33m(L)];
disp(T_final)

disp('mode relaxation stress finals T11 T12 T22')
disp(tau_finalm)

disp('relaxation final values for a11 a12 a22 a33 =')
a_final = [a11m(L) a12m(L) a22m(L) a33m(L)]';
disp(a_final)


%%%%This section computes the fiber stress from Equation (3)
if r~=0
    error('r~=0')   %relaxation period so r shold =0
else
    disp('r==0')
    tf12m=(2.0*phi.*total12m.*Ap).*((27.0*(a11m.*a22m.*(1-a11m-a22m)-(a12m.^2).*(1-a11m-a22m))).*(-
1.0/35.0+1.0/7.0*(a11m+a22m))+(a12m.^2).*(1-27*(a11m.*a22m.*(1-a11m-a22m))+27.0*(a12m.^2).*(1-a11m-
a22m)));
    tf11m=(2.0*phi.*total12m.*Ap).*((27.0*(a11m.*a22m.*(1-a11m-a22m)-(a12m.^2).*(1-a11m-
a22m))).*(3.0/7.0*(a12m))+(a11m.*a12m).*(1-27*(a11m.*a22m.*(1-a11m-a22m))+27.0*(a12m.^2).*(1-a11m-
a22m)));
    tf22m=(2.0*phi.*total12m.*Ap).*((27.0*(a11m.*a22m.*(1-a11m-a22m)-(a12m.^2).*(1-a11m-
a22m))).*(3.0/7.0*(a12m))+(a22m.*a12m).*(1-27*(a11m.*a22m.*(1-a11m-a22m))+27.0*(a12m.^2).*(1-a11m-
a22m)));
    tf33m=(2.0*phi.*total12m.*Ap).*((27.0*(a11m.*a22m.*(1-a11m-a22m)-(a12m.^2).*(1-a11m-
a22m))).*(1.0/7.0*(a12m))+((1-a11m-a22m).*a12m).*(1-27*(a11m.*a22m.*(1-a11m-a22m))+27.0*(a12m.^2).*(1-
a11m-a22m)));
```

```
end

tauc12m=tf12m+total12m;
tauc11m=tf11m+total11m;
tauc22m=tf22m+total22m;
tauc33m=tf33m+total33m;

end

%-------------------------------------------------------------------------
%   REVERSE  DIRECTION
%-------------------------------------------------------------------------
tau_finalr=zeros(3,modes);
r=-r1;

for x=1:modes
   if resttime~=0
      initialr=[tau_finalm(:,x); a_final];
   else
      initialr=[tau_finalf(:,x); a_final];
   end
      tspanr=time1(end)+resttime:0.01:time1(end)+resttime+shear_time;
   [timer, yo]=ode23tb(@modepolymersub,tspanr,initialr);
   export = [' mode ' num2str(x) ' reverse calculation done'];
   disp(export)

   L=length(timer);

   if x==1
      tau1=zeros(L,modes);
      tau12=zeros(L,modes);
      tau2=zeros(L,modes);
      tau3=zeros(L,modes);
   end

   tau1(:,x)=yo(:,1);
   tau12(:,x)=yo(:,2);
   tau2(:,x)=yo(:,3);
   tau3(:,x)=yo(:,4);


   tau_finalr(:,x) = [tau1(L,x) tau12(L,x) tau2(L,x) tau3(L,x)]';

end

total11r=sum(tau1,2);
total12r=sum(tau12,2);
total22r=sum(tau2,2);
total33r=sum(tau3,2);

a11r=yo(:,5);
a12r=yo(:,6);
a22r=yo(:,7);
```

```
a33r=1-a11r-a22r;

disp('final reverse values for T11p T12p T22p T33p')
T_final= [total11r(L) total12r(L) total22r(L) total33r(L)];
disp(T_final)

disp('mode reverse stress finals T11 T12 T22')
disp(tau_finalr)

disp('reverse final values for a11 a12 a22 a33 =')
a_final = [a11r(L) a12r(L) a22r(L) a33r(L)];
disp(a_final)

etar=total12r./r;
coef=2.0*etar*phi;

%%%%This section computes the fiber stress from Equation (3)
if r~=0
   disp('r~=0')
   tf12r=(coef*Ap*r).*((27.0.*(a11r.*a22r.*(1-a11r-a22r)-(a12r.^2).*(1-a11r-a22r))).*(-
1.0/35.0+1.0/7.0.*(a11r+a22r))+(a12r.^2).*(1-27.*(a11r.*a22r.*(1-a11r-a22r))+27.0.*(a12r.^2).*(1-a11r-a22r)));
   tf11r=(coef*Ap*r).*((27.0.*(a11r.*a22r.*(1-a11r-a22r)-(a12r.^2).*(1-a11r-
a22r))).*(3.0/7.0.*(a12r))+(a11r.*a12r).*(1-27*(a11r.*a22r.*(1-a11r-a22r))+27.0.*(a12r.^2).*(1-a11r-a22r)));
   tf22r=(coef*Ap*r).*((27.0.*(a11r.*a22r.*(1-a11r-a22r)-(a12r.^2).*(1-a11r-
a22r))).*(3.0/7.0.*(a12r))+(a22r.*a12r).*(1-27*(a11r.*a22r.*(1-a11r-a22r))+27.0.*(a12r.^2).*(1-a11r-a22r)));
   tf33r=(coef*Ap*r).*((27.0.*(a11r.*a22r.*(1-a11r-a22r)-(a12r.^2).*(1-a11r-a22r))).*(1.0/7.0.*(a12r))+((1-a11r-
a22r).*a12r).*(1-27.*(a11r.*a22r.*(1-a11r-a22r))+27.0.*(a12r.^2).*(1-a11r-a22r)));
else
   error('r==0')

end

tauc12r=tf12r+total12r;
tauc11r=tf11r+total11r;
tauc22r=tf22r+total22r;
tauc33r=tf33r+total33r;
etacr=tauc12r./r;

%--------------------------------------------------------------------------
%   COMBINE AND GRAPH
%--------------------------------------------------------------------------

if resttime~=0
   tauc12=[tauc12f;tauc12m;tauc12r];   %combines stress from each section into a single vector
   tauc11=[tauc11f;tauc11m;tauc11r];
   tauc22=[tauc22f;tauc22m;tauc22r];
   etac=[etacf;etacm;etacr];
   time=[time1;time2;timer];
else
   time=[time1;timer-timer(1)+time1+0.01];
end


timex=timer-time1(end)-resttime;
```

```
%%%%Data section categorically named so it can be called upon automatically
%%%%to match conditions that were modeled

    %Data From Will 2 wt%, r=1, forward only
    timedataf1 = [0.01 0.02 0.03 0.04 0.05 0.06 0.07 0.08 0.09 0.1 0.11 0.12 0.13 0.14
0.15 0.16 0.17 0.185 0.215 0.255 0.3 0.355 0.425 0.505 0.6 0.715 0.85 1.01 1.2 1.43 1.7 2.02
2.4 2.855 3.4 4.045 4.81 5.72 6.805 8.095 9.63 11.46 13.635 16.22 19.3 22.965 27.32 32.505
38.675 46.015];
    expdataf21 = [54.8317 564.6663333 1257.98 1978.613333 2655.493333 3267.576667 3802.68
4298.746667 4744.333333 5152.246667 5536.08 5884.146667 6214.066667 6516.61 6820.003333 7107.283333
7368.97 7752.85 8507.93 9358.323333 10144.71333 11012.05 11974.76667 12890.3 13796.4 14726.23333
15630.16667 16440.73333 17210.43333 17887.6 18451.5 18860.06667 19119.16667 19203.76667 19163.23333
18983.16667 18714.63333 18396.2 18052.6 17738.16667 17444.53333 17192.9 16967.2 16771.6 16596.1
16468.86667 16323.2 16125.26667 15915.56667 15721.53333];

    %Data From Will 5 wt%, r=1, forward only
    timedataf1 = [0.01 0.02 0.03 0.04 0.05 0.06 0.07 0.08 0.09 0.1 0.11 0.12 0.13 0.14
0.15 0.16 0.17 0.185 0.215 0.255 0.3 0.355 0.425 0.505 0.6 0.715 0.85 1.01 1.2 1.43 1.7 2.02
2.4 2.855 3.4 4.045 4.81 5.72 6.805 8.095 9.63 11.46 13.635 16.22 19.3 22.965 27.32 32.505
38.675 46.015];
    expdataf51 = [87.22605 875.2295 1989.45 3172.08 4273.23 5271.7 6163.31 6961.58 7683.185
8350.365 8951.785 9512.44 10054.975 10563.76 11063.475 11489.685 11917.935 12530.3 13649.75
14981.8 16216.2 17500.4 18959.9 20359.35 21705.2 23070.55 24374.5 25565.85 26666.1 27619.25 28400.3
28852 29054.65 28968.85 28591.4 27980.15 27205.4 26320.2 25428.35 24608.75 23847.75 23171.3
22580.45 22025.15 21550 21112.65 20729.8 20377.9 19890.15 19364.75];

        %Data From Will 2 wt%, r=0.1, forward only
    timedatafp = [0.01 0.02 0.03 0.04 0.05 0.06 0.07 0.08 0.09 0.1 0.11 0.12 0.135 0.165
0.205 0.25 0.31 0.39 0.485 0.6 0.75 0.935 1.165 1.455 1.81 2.255 2.81 3.5 4.365 5.445 6.79
8.465 10.555 13.165 16.42 20.475 25.53 31.835 39.705 49.515 61.745 77.005 106.585 132.925 165.765
206.725 257.805 321.505 400.935 500.005];
    expdataf2p = [-28.60496667 443.5626667 1193.442667 1972.15 2735.973333 3403.826667 4034.83
4573.966667 5075.016667 5535.47 5957.303333 6345.503333 6907.106667 7933.293333 9092.84 10215.42667
11497.3 12954 14398.53333 15846.76667 17406.13333 18954.4 20514.06667 22172.6 23974.9 25684.93333
27097.3 28527.1 29875.2 30990.9 32182.93333 33062.03333 33815.83333 34364.9 34754.5 34892.8 34841.2
34631.63333 34292.76667 33907.5 33501.1 33124.16667 32801.43333 32584.96667 32437.56667 32364.1
32258.66667 32269.73333 32245.23333 32114.66667];

    %Data From Will 5 wt%, r=0.1, forward only
    timedatafp = [0.01 0.02 0.03 0.04 0.05 0.06 0.07 0.08 0.09 0.1 0.11 0.12 0.135 0.165
0.205 0.25 0.31 0.39 0.485 0.6 0.75 0.935 1.165 1.455 1.81 2.255 2.81 3.5 4.365 5.445 6.79
8.465 10.555 13.165 16.42 20.475 25.53 31.835 39.705 49.515 61.745 77.005 106.585 132.925 165.765
206.725 257.805 321.505 400.935 500.005];
    expdataf5p = [54.64783333 768.4926667 1888.563333 3135.736667 4344.343333 5384 6359.803333
7229.013333 8050.643333 8755.083333 9470.336667 10089.49333 10986.26667 12573.26667 14378.86667
16131.1 18117.26667 20326.36667 22532.76667 24705.8 27045.66667 29492.23333 31878.5 34198.03333
36397.93333 38585.5 40327.9 42087.03333 43897.66667 45157.33333 46242.23333 47230.56667 47865.26667
48410.3 48647.3 48690.86667 48356.6 47634.93333 46584.53333 45316.06667 43983.4 42792.23333 41847.46667
41065.13333 40611.8 40329.6 40256.2 40542.3 40904.36667 40874.16667];

    %Data From Will 2 wt%, r=0.1, rest time=0, reverse
    timedatarp = [500.01 500.02 500.03 500.04 500.05 500.06 500.07 500.08 500.09 500.1 500.11
500.12 500.135 500.165 500.205 500.25 500.31 500.39 500.485 500.6 500.75 500.935 501.165 501.455 501.81
```

502.255 502.81 503.5 504.365 505.445 506.79 508.465 510.555 513.165 516.42 520.475 525.53 531.835 539.705 549.515 561.745 577.005 606.585 632.925 665.765 706.725 757.805 821.505 900.935 1000.005];

expdatar2p0 = [-29422 -28640 -27333 -26013 -24798 -23712 -22605 -21871 -21050 -20303 -19587 -18978 -18027 -16402 -14468 -12653 -10596 -8178.4 -5827.4 -3399.2 -772.5 1863.37 4482.17 7168.12 9813.94 12493 14848.3 17239.2 19497.3 21630.7 23532.1 25143.1 26648.8 27791.7 28784.8 29511.5 30061.3 30378.3 30588.8 30680.7 30680.4 30666.9 30645 30642.7 30626.6 30746.6 30723.2 30652.1 30765.8 30739.2];

%Data From Will 2 wt%, r=0.1, rest time=20, reverse

expdatar2p20 = [-1226.1 -785.51 -85.109 679.601 1388.2 2038.59 2608.01 3118.06 3569.61 4009.09 4412.49 4799.2 5321.97 6291.05 7377.11 8456.91 9670.26 10996.8 12357.4 13757.4 15289 16888.1 18504.7 20240.8 21823.8 23212 24774.7 26288.3 27719.7 29078.8 30160.6 31217.1 32114.9 32846.1 33367.6 33811.3 34070.8 34222.6 34285.3 34278.1 34229.4 34167.2 34127.3 34085.3 34064.9 34077.9 34141.4 34042.9 33994.1 33822.2];

%Data From Will 2 wt%, r=0.1, rest time=400, reverse

expdatar2p400 = [39.3117 445.593 1122.19 1829.5 2462.48 3075.6 3616.72 4088.46 4547.81 4988.4 5386.87 5740.52 6207.98 7108.95 8151.62 9169.07 10326.5 11657.9 12974 14325.7 15825.5 17416.8 18968.8 20427.2 21846.3 23392.3 24909.5 26336.3 27682.4 28954.3 29983.5 31003.5 31689.8 32321 32750 33001 33128.1 33135.5 33062.5 32931.9 32755.6 32619.7 32511.2 32430.7 32368 32332.9 32233.4 32110.3 32048.6 31856];

%Data From Will 2 wt%, r=1, rest time=0, reverse

timedatar1 = [50.01 50.02 50.03 50.04 50.05 50.06 50.07 50.08 50.09 50.1 50.11 50.12 50.13 50.14 50.15 50.16 50.17 50.185 50.215 50.255 50.3 50.355 50.425 50.505 50.6 50.715 50.85 51.01 51.2 51.43 51.7 52.02 52.4 52.855 53.4 54.045 54.81 55.72 56.805 58.095 59.63 61.46 63.635 66.22 69.3 72.965 77.32 82.505 88.675 96.015];

expdatar210 = [-15239 -14525 -13370 -12254 -11205 -10290 -9487.7 -8726.8 -8078.3 -7488.3 -6948.7 -6445.2 -5967.8 -5537.5 -5114.4 -4707.5 -4311 -3775.8 -2785.3 -1612.6 -489.6 703.18 1992.16 3273.77 4486.08 5774.65 7023.2 8207.52 9384.83 10514.7 11535.5 12459 13260.4 13950.3 14511.3 14920.2 15169.5 15268.3 15305.8 15293.3 15230.5 15182.6 15113.8 15044.3 14997.1 14936.4 14913 14806.1 14730.9 14547];

%Data From Will 2 wt%, r=1, rest time=20, reverse

expdatar2120 = [-80.107 305.417 838.14 1399.41 1912.86 2416.13 2826.59 3235.9 3583.41 3903.97 4186.24 4488.61 4759.96 5031.56 5281.55 5522.38 5736.5 6056.81 6612.53 7291.54 7987.44 8716.92 9526.66 10323.6 11121.2 11903 12664.4 13386.6 14039.9 14612.2 15095.2 15449.1 15691.1 15817.7 15821.2 15709.8 15503.8 15247.1 14981.9 14804 14635.1 14504.4 14317.9 14123 13842.4 13529.6 13244 13176.2 13488.9 13897.3];

%Data From Will 2 wt%, r=1, rest time=400, reverse

expdatar21400 = [150.635 603.966 1251.89 1934.97 2575.95 3155.9 3684.74 4170.45 4577.09 4979.15 5338.75 5667.57 5981.13 6274.35 6557.39 6821.11 7089.41 7468.58 8159.16 8990.24 9783.55 10670.9 11626.7 12584.7 13481.6 14434.9 15359.6 16208 17003.5 17730.8 18327.7 18756.1 19067.9 19214.7 19224.6 19107.9 18909.3 18645.1 18376.1 18114.6 17891.6 17741.5 17615 17489.4 17362.7 17252.4 17134.7 17075.6 16947.5 16680.3];

%Data From Will 5 wt%, r=0.1, rest time=0, reverse

timedatarp = [500.01 500.02 500.03 500.04 500.05 500.06 500.07 500.08 500.09 500.1 500.11 500.12 500.135 500.165 500.205 500.25 500.31 500.39 500.485 500.6 500.75 500.935 501.165 501.455 501.81 502.255 502.81 503.5 504.365 505.445 506.79 508.465 510.555 513.165 516.42 520.475 525.53 531.835 539.705 549.515 561.745 577.005 606.585 632.925 665.765 706.725 757.805 821.505 900.935 1000.005];

```
expdatar5p0 = [-44511  -43547  -41853  -40152  -38458  -36973  -35503  -34389  -33072  -32172  -31172  -30212  -28920  -26727  -24126  -21561  -18596  -15292  -11971  -8552.3 -4788.2 -891.38 3051.97 7115.36 11041.1 14955.1 18859.9 22521.2 26043.9 29189.3 32160.2 34798.3 37007.6 38899.6 40390.5 41512.2 42333.4 42824.7 43110  43222.8 43254.3 43256.8 43260  43294.1 43377.3 43453.1 43554  43635.3 43762.7 43600.4];

%Data From Will 5 wt%, r=0.1, rest time=20, reverse
expdatar5p20 = [-1575.4 -1128.5 -445.58 314.253 1016.6  1654.24 2215.37 2762.29 3220.52 3672.55 4066.89 4469.49 4968.89 5894.98 6969.6  8048.37 9276.97 10668.2 12086.8 13594.3 15221.1 16896.4 18592.9 20294  21916.5 23689.8 25359.4 27000.3 28377.5 29761  31058.6 32125.7 33052.5 33797.7 34385.3 34799.3 35085.2 35223.3 35309.5 35323.1 35313.9 35307.5 35334.1 35288.6 35299.8 35297.2 35213.7 35184.4 35445.3 36130.1];

%Data From Will 5 wt%, r=0.1, rest time=400, reverse
expdatar5p400 = [27.9889  533.675 1398.34 2312.54 3187.68 3986.05 4693.82 5351.69 5914.99 6452.86 6982.14 7438.94 8077.58 9255.19 10598.4 11882.3 13369.6 15124.6 16777.7 18539.1 20423.1 22336.4 24152.5 26046.5 27921  29933.3 31520.6 33115.9 34712.3 36147.9 37419  38339.2 39223.6 39750.4 40213.8 40470.6 40617.7 40624.4 40620.8 40554.7 40497.2 40481.8 40530.1 40582.4 40669.8 40746.4 40922.7 41040  41349.5 41392];

%Data From Will 5 wt%, r=1, rest time=0, reverse
timedatar1=[50.01  50.02  50.03  50.04  50.05  50.06  50.07  50.08  50.09  50.1  50.11  50.12  50.13 50.14  50.15  50.16  50.17  50.185  50.215  50.255  50.3  50.355  50.425  50.505  50.6  50.715  50.85  51.01 51.2  51.43  51.7  52.02  52.4  52.855  53.4  54.045  54.81  55.72  56.805  58.095  59.63  61.46  63.635 66.22  69.3  72.965  77.32  82.505  88.675  96.015];

expdatar510 = [-18680  -17884  -16476  -15271  -14034  -12993  -12049  -11155  -10417  -9708  -9066.1 -8457.8 -7897.3 -7393.8 -6871.9 -6392.3 -5944.2 -5294.7 -4086.5 -2681.8 -1326.9 121.433 1693.95 3276.96 4768.93 6360.23 7900.76 9418.35 10852.5 12142.3 12786.8 13017.7 13478.2 14121.6 15896.9 17581.2 18274.6 18570.7 18619.1 18541.6 18410.8 18222.9 18088.8 18076.6 18097.4 18063.2 18019.3 18060.6 17973.7 17884.4];

%Data From Will 5 wt%, r=1, rest time=20, reverse
expdatar5120 = [-192.16 270.471 967.685 1684.56 2355.74 2970.66 3533.19 4042.83 4537.7  4962.65 5427.52 5770.33 6144.42 6492.31 6813  7129.61 7397.65 7800.1  8575.99 9506.51 10451.4 11430.4 12516.6 13597.4 14622.8 15674  16697.7 17627  18489.4 19239.6 19897.8 20367  20666.5 20777.1 20749.9 20603 20353.4 19954.6 19521.8 19130.7 18819.6 18487.7 18166.7 17888.2 17724.7 17580.8 17305.3 16775  16199.1 15827.4];

%Data From Will 5 wt%, r=1, rest time=400, reverse
expdatar51400 = [224.512  696.655 1364.99 2045.32 2683.89 3258.89 3761.66 4219.15 4637.74 5023.27 5391  5748.56 6051.95 6373.39 6674.75 6946.86 7224.56 7609.71 8299.53 9116.11 9944.7  10788.7 11749.4 12690.4 13577.3 14516.7 15420.1 16280.7 17043.8 17772  18348.7 18782.2 19099.4 19268.1 19288.5 19191.3 19015.6 18782.8 18535.8 18303.2 18114.2 17973.1 17855.4 17751.6 17661.3 17585.9 17516.2 17359.5 17168.6 16968.6];

if r1==1
   rr='1';
elseif r1==0.1
   rr='p';
else
   disp('no data')
end


%%%%Chooses which data corresponds with situation that is modeled
```

```
 timedataf=eval(['timedataf' rr]);
 timedatar=eval(['timedatar' rr]);
 expdataf=eval(['expdataf' num2str(massfrac*100) rr]);
 expdatar=eval(['expdatar' num2str(massfrac*100) rr num2str(resttime)]);


%%%Graphing section for forward and reverse flows against experimental data
graph=1;
if graph==1

    figure(1);
    semilogx(time1,etacf,timedataf,expdataf)
    legend('etacf model','etacf exp')
    title('Forward')
    hold on

    figure(2);
    semilogx(timer-timer(1)+0.01,etacr,timedatar-timedatar(1)+0.01,expdatar)
    legend('etacr model','etacr exp')
    title('Reverse')
    hold on
end

%for export from program
model=[time1 etacf etacr];
experiment=[timedataf' expdataf' expdatar'];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%
%Function that the ode solvers call to solve Equations (2) and (4) simultaneously
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%
    function dyo = modepolymersub(t,y)

        tp11=y(1);
        tp12=y(2);
        tp22=y(3);
        tp33=y(4);
        a11=y(5);
        a12=y(6);
        a22=y(7);
        a33=1-a11-a22;
        dyo=zeros(7,1);

        dyo(1,1)=-alpha(x)*(tp11^2+tp12^2)/etap(x)-sigma*tp11/lambda(x)+2*r*tp12-3*(1-
sigma)*(tp11*a11+tp12*a12)/lambda(x);
        dyo(2,1)=-alpha(x)*(tp11*tp12+tp12*tp22)/etap(x)-sigma*tp12/lambda(x)+etap(x)*r/lambda(x)+r*tp22-3*(1-
sigma)/2/lambda(x)*(a11*tp12+a12*tp22+a12*tp11+a22*tp12);
        dyo(3,1)=-alpha(x)*(tp12^2+tp22^2)/etap(x)-sigma*tp22/lambda(x)-3*(1-
sigma)/lambda(x)*(tp12*a12+tp22*a22);
        dyo(4,1)=-alpha(x)*(tp33^2)/etap(x)-tp33*sigma/lambda(x)-3*(1-sigma)/lambda(x)*a33*tp33;

        dyo(5,1)=r*a12+2*CI*abs(r)*(1.0-3.0*a11)+chi*r*a12-...
          2*chi*r*(3.0/7.0*a12*(27.0*a11*a22*(1-a11-a22)...
```

```
        -27.0*(a12^2)*(1-a11-a22))+(1.0-27.0*a11*a22*(1-a11-a22)+...
        27.0*(a12^2)*(1-a11-a22))*a11*a12);
    dyo(6,1)=1.0/2.0*r*a22-1.0/2.0*r*a11+chi*((1.0/2.0*r*a22+...
        1.0/2.0*r*a11)-(2.0*r)*((27.0*a11*a22*(1-a11-a22)-...
        27.0*(a12^2)*(1-a11-a22))*(-1.0/35.0+1.0/7.0*a11+1.0/7.0*a22)+...
        (1.0-27.0*a11*a22*(1-a11-a22)...
        +27.0*(a12^2)*(1-a11-a22))*a12^2))-6.0*CI*abs(r)*a12;
    dyo(7,1)=-r*a12+2*CI*abs(r)*(1.0-3.0*a22)+chi*r*a12-...
        2*chi*r*(3.0/7.0*a12*(27.0*a11*a22*(1-a11-a22)-...
        27.0*(a12^2)*(1-a11-a22))+(1.0-27.0*a11*a22*(1-a11-a22)+...
        27.0*(a12^2)*(1-a11-a22))*a12*a22);

  end

end
```

Appendix B: MATLAB program used for optimizing η_p, λ and α based on transient viscosity and SAOS experiments for the pure polymer

```
function Final = PolymerFit

%This program attempts to find the best eta, lambda and alpha values to fit
%both transient viscosity and G' and G" data. Both sets of data are
%calculated based on eta, lambda and alpha values then error is calculated,
%scaled and added together. The optimization function "fmincon" attempts to
%minimize this total summed error by adjusting eta, lambda and alpha values

format long


%initial guesses
Eta0=[751.7839  2920.1549   9277.66138  14316.446   7408.894];
lambda=[0.007775542 0.070987006 0.474159253 2.982311897 18.16897262];
alpha=[0.9999   0.7 0.7 0.7 0.583723836];

%frequencies tested in G' and G" determination
freq=[0.01585   0.02512 0.03981 0.0631  0.1 0.15849 0.25119 0.39811 0.63096 1   1.58489 2.51189 3.98107
6.30957 10  15.8489 25.1189 39.8107 63.0957 100];

%experimental G' and G" values
Gexp=[35.06125  118.77715   218.069 403.069 731.4956667 1295.72 2281.803333 3723.183333 5670.693333
8656.983333 12385.93333 17431.8 23216.63333 30205.2 38639.13333 47962.26667 58520.46667 70267.56667
82679.76667 96278.46667];
GDexp=[532.1383333  799.4886667 1259.376667 1859.03 2733.77 3907.66 5450.736667 7393.43 9767.963333
12548.53333 15681.63333 19139.5 22729.5 26576.8 30379.83333 34434.86667 38387.86667 42793.26667 47297.4
52348.03333];

%choice of algorithm
options=optimset('Algorithm','interior-point');

f0=[Eta0; lambda; alpha];

%call to optimization function
[Final fval exitflag]=fmincon(@FIT,f0,[],[],[],[],[0 0 0 0 0;0 0 0 0 0;0 0 0 0 0],[inf inf inf inf inf;inf inf inf inf
inf;0.9999 0.9999 0.9999 0.9999 0.9999],[],options)

%-------------------------------------------------------------------------
%Function "FIT" takes parameters guessed by "fmincon" and adjusts values so
%that the simulation will converge if needed. Transient viscosity is
%calculated by sending conditions to "PolymerFitSimul". G'and G" are
%calculated based on the guessed parameters and error calculated
%based on experimental data. Errors are scaled and summed before being sent
%back to "fmincon"
%-------------------------------------------------------------------------
function e = FIT(para)
   Gerror=0;
   Gprime=zeros(length(freq),1);
   GDprime=zeros(length(freq),1);
```

```
Eta=para(1,:);
Lambda=para(2,:);
Alpha=para(3,:);

for g=1:5
   if Lambda(g)>=0.03 && Lambda(g)<=0.04 && Alpha(g)>0.84
      Alpha(g)=0.84;
   elseif Lambda(g)>0.04 && Lambda(g)<=1 && Alpha(g)>0.68
      Alpha(g)=0.68;
   elseif Lambda(g)>1 && Alpha(g)>0.7
      Alpha(g)=0.7;
   end
end



for j=1:length(freq)
      Gtemp=zeros(1,5);
      GDtemp=zeros(1,5);
   for i=1:5
      Gtemp(i)=Eta(i)*Lambda(i)*freq(j)^2/(1+(Lambda(i)*freq(j))^2);
      GDtemp(i)=Eta(i)*freq(j)/(1+(Lambda(i)*freq(j))^2);
   end

   Gprime(j)=sum(Gtemp);
   GDprime(j)=sum(GDtemp);

   Gerror=Gerror+(log10(Gexp(j))-log10(Gprime(j)))^2+(log10(GDexp(j))-log10(GDprime(j)))^2;

end

Terror=zeros(1,2);

for i=1:2

   if i==1
      r=0.1;   %Shear rate
      wt=0;   %mass fraction CNFs


      % Data From:
      timedata = [0.01   0.02   0.03   0.04   0.05   0.06   0.07   0.08   0.09   0.1 0.11   0.12   0.135   0.165
0.205   0.25   0.31   0.39   0.485   0.6 0.75   0.935   1.165   1.455   1.81   2.255   2.81   3.5 4.365   5.445   6.79
8.465   10.555  13.165  16.42   20.475  25.53   31.835  39.705  49.515  61.745  77.005  106.585 132.925 165.765
206.725 257.805 321.505 400.935 500.005];
      expdata = [-24.58123333 359.0053333 994.5776667 1631.24 2260.15 2811.76 3320.696667 3759.223333
4193.453333 4587.953333 4953.843333 5266.38 5745.383333 6594.616667 7558.21 8531.79 9642.793333
10907.93333 12176.2 13471.9 14867.76667 16285.23333 17695.83333 19167.5 20791.23333 22509.83333
23954.36667 25375.7 26772.8 27998.4 29265.66667 30267.96667 31109.2 31775.73333 32261.03333 32519.56667
32572.26667 32497.46667 32261.3 31940.33333 31590.56667 31247.83333 30907.5 30585.33333 30310   30083.3
29869.83333 29742.9 29607.03333 29358.7];
   elseif i==2
```

```
        timedata = [0.01    0.02    0.03    0.04    0.05    0.06    0.07    0.08    0.09    0.1 0.11    0.12    0.13    0.14
0.15    0.16    0.17    0.185   0.215   0.255   0.3 0.355   0.425   0.505   0.6 0.715   0.85    1.01    1.2 1.43    1.7 2.02
2.4 2.855   3.4 4.045   4.81    5.72    6.805   8.095   9.63    11.46   13.635  16.22   19.3    22.965  27.32   32.505
38.675  46.015];
          expdata = [53.8919    410.7393333 899.2103333 1421.429333 1907.116667 2352.536667 2752.906667
3118.393333 3451.083333 3748.77 4028.253333 4317.403333 4576.376667 4814.62 5043.283333 5273.18 5489.98
5804.03 6394.62 7103.683333 7787.536667 8543.646667 9414.366667 10281.52333 11175.45333 12119.92333
13048.74    13986.03667 14923.03333 15824.96667 16617.56667 17291.83333 17797.2 18121    18258.63333
18222.73333 18092.53333 17858.16667 17580.26667 17304.4 17050.13333 16825.13333 16645.56667
16409.83333 16122.3 15774.43333 15492.43333 15247.53333 14979.3 14742.66667];
          r=1;
      end

   etacmod=zeros(length(timedata),1);


   timedata=timedata*100;
   timedata=round(timedata);
   timedata=timedata/100;

   [t etatemp]=PolymerFitSimul(r,wt,Eta,Lambda,Alpha,timedata);

   t=t*100;
   t=round(t);
   t=t/100;


        for k=1:length(timedata)
           Terror(i)=Terror(i) + (expdata(k)-etatemp(k))^2;
        end
   end

        Terror=sum(Terror);

        Terror=Terror/1.1E8;
        Gerror=Gerror/0.8;

        para
        e=Terror+Gerror

end

end
```

The below program, “PolymerFitSimul”, solves the model and sends results back to the main function “PolymerFit”.

```
%Original shear program written By Monon Mahboob and Michael Smith
%Edited by William Murch Tim Kremer
```

```
%Solves equations and sends results back to polymer parameter fitting
%optimization program "PolymerFit"

function [time etac] = PolymerFitSimul(r,massfrac,etap,lambda,alpha,tspan)

x=1;
sigma=1;
CI=0.09;    %Does not affect simulation of pure polymer

re=33.0; %aspect ratio

chi=1.0*(re^2-1)/(re^2+1); %another form of aspect ratio
rf=1750.0; %fiber density
rs=1000.0; %polymer density


modes=5;

phi=1.0*rs*massfrac/(rf+(rs-rf)*massfrac);  %volume fraction
Ap=1.0*re^2/(3*log(sqrt(1.0*pi/phi)));  %A2, shape factor

orientation=[0.333, 0.0, 0.333];    %initially random orientation

tau_final=zeros(3,modes);

for x=1:modes
   initial=[0 0 0 0 orientation];  %initial conditions
   [time, yo]=ode23tb(@modepolymersub,tspan,initial);  %ode solver
   export = [' mode ' num2str(x) ' calculation done'];
   disp(export)

   L=length(time);

   if x==1
      tau1=zeros(L,modes);
      tau12=zeros(L,modes);
      tau2=zeros(L,modes);
      tau3=zeros(L,modes);
   end

   tau1(:,x)=yo(:,1);
   tau12(:,x)=yo(:,2);
   tau2(:,x)=yo(:,3);
   tau3(:,x)=yo(:,4);


   tau_final(:,x) = [tau1(L,x) tau12(L,x) tau2(L,x)]';

end

total11=sum(tau1,2);
total12=sum(tau12,2);
```

```
total22=sum(tau2,2);
total33=sum(tau3,2);

a11=yo(:,5);
a12=yo(:,6);
a22=yo(:,7);
a33=1-a11-a22;

eta=total12./r;
coef=2.0*eta*phi;

%%%%This section computes the fiber stress from Equation (3)
if r~=0
   tf12=(coef*Ap*r).*((27.0.*(a11.*a22.*(1-a11-a22)-(a12.^2).*(1-a11-a22))).*(-
1.0/35.0+1.0/7.0.*(a11+a22))+(a12.^2).*(1-27.*(a11.*a22.*(1-a11-a22))+27.0.*(a12.^2).*(1-a11-a22)));
else
   tf12=(2.0*phi.*total12.*Ap).*((27.0*(a11.*a22.*(1-a11-a22)-(a12.^2).*(1-a11-a22))).*(-
1.0/35.0+1.0/7.0*(a11+a22))+(a12.^2).*(1-27*(a11.*a22.*(1-a11-a22))+27.0*(a12.^2).*(1-a11-a22)));
end

tauc12=tf12+total12;
etac=tauc12./r;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%
%Function that the ode solvers call to solve Equations (2) and (4) simultaneously
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%
   function dyo = modepolymersub(t,y)

       tp11=y(1);
       tp12=y(2);
       tp22=y(3);
       tp33=y(4);
       a11=y(5);
       a12=y(6);
       a22=y(7);
       a33=1-a11-a22;
       dyo=zeros(7,1);

       dyo(1,1)=-alpha(x)*(tp11^2+tp12^2)/etap(x)-sigma*tp11/lambda(x)+2*r*tp12-3*(1-
sigma)*(tp11*a11+tp12*a12)/lambda(x);
       dyo(2,1)=-alpha(x)*(tp11*tp12+tp12*tp22)/etap(x)-sigma*tp12/lambda(x)+etap(x)*r/lambda(x)+r*tp22-3*(1-
sigma)/2/lambda(x)*(a11*tp12+a12*tp22+a12*tp11+a22*tp12);
       dyo(3,1)=-alpha(x)*(tp12^2+tp22^2)/etap(x)-sigma*tp22/lambda(x)-3*(1-
sigma)/lambda(x)*(tp12*a12+tp22*a22);
       dyo(4,1)=-alpha(x)*(tp33^2)/etap(x)-tp33*sigma/lambda(x)-3*(1-sigma)/lambda(x)*a33*tp33;

       dyo(5,1)=r*a12+2*CI*abs(r)*(1.0-3.0*a11)+chi*r*a12-...
          2*chi*r*(3.0/7.0*a12*(27.0*a11*a22*(1-a11-a22)...
          -27.0*(a12^2)*(1-a11-a22))+(1.0-27.0*a11*a22*(1-a11-a22)+...
          27.0*(a12^2)*(1-a11-a22))*a11*a12);
       dyo(6,1)=1.0/2.0*r*a22-1.0/2.0*r*a11+chi*((1.0/2.0*r*a22+...
          1.0/2.0*r*a11)-(2.0*r)*((27.0*a11*a22*(1-a11-a22)-...
```

```
        27.0*(a12^2)*(1-a11-a22))*(-1.0/35.0+1.0/7.0*a11+1.0/7.0*a22)+...
        (1.0-27.0*a11*a22*(1-a11-a22)...
        +27.0*(a12^2)*(1-a11-a22))*a12^2))-6.0*CI*abs(r)*a12;
    dyo(7,1)=-r*a12+2*CI*abs(r)*(1.0-3.0*a22)+chi*r*a12-...
        2*chi*r*(3.0/7.0*a12*(27.0*a11*a22*(1-a11-a22)-...
        27.0*(a12^2)*(1-a11-a22))+(1.0-27.0*a11*a22*(1-a11-a22)+...
        27.0*(a12^2)*(1-a11-a22))*a12*a22);

  end

end
```

Appendix C: MATLAB programs used for calculating error in transient shear viscosity associated with different combinations of σ and C_I in nanocomposite forward transient shear flows.

```
function error = sigmaCIForwardFit

sigma=0.6:0.02:1;  %Sigmas to test
CI=0.01:0.002:0.1;      %CIs to test



for x=1:4

   if x==1
        %Data From Will 2 wt%, r=1, forward only
        timedata = [0.01    0.02    0.03    0.04    0.05    0.06    0.07    0.08    0.09    0.1 0.11    0.12    0.13    0.14
0.15    0.16    0.17    0.185   0.215   0.255   0.3 0.355   0.425   0.505   0.6 0.715   0.85    1.01    1.2 1.43    1.7 2.02
2.4 2.855   3.4 4.045   4.81    5.72    6.805   8.095   9.63    11.46   13.635  16.22   19.3    22.965  27.32   32.505
38.675  46.015];
        expdata = [54.8317  564.6663333 1257.98 1978.613333 2655.493333 3267.576667 3802.68 4298.746667
4744.333333 5152.246667 5536.08 5884.146667 6214.066667 6516.61 6820.003333 7107.283333 7368.97 7752.85
8507.93 9358.323333 10144.71333 11012.05    11974.76667 12890.3 13796.4 14726.23333 15630.16667
16440.73333 17210.43333 17887.6 18451.5 18860.06667 19119.16667 19203.76667 19163.23333 18983.16667
18714.63333 18396.2 18052.6 17738.16667 17444.53333 17192.9 16967.2 16771.6 16596.1 16468.86667 16323.2
16125.26667 15915.56667 15721.53333];
        r=1;   %Shear rate
        wt=0.02;

        etacmod=zeros(length(timedata),length(sigma),length(CI));
        error=zeros(length(sigma),length(CI));


      elseif x==2
        x
        %Data From Will 5 wt%, r=1, forward only
        timedata = [0.01    0.02    0.03    0.04    0.05    0.06    0.07    0.08    0.09    0.1 0.11    0.12    0.13    0.14
0.15    0.16    0.17    0.185   0.215   0.255   0.3 0.355   0.425   0.505   0.6 0.715   0.85    1.01    1.2 1.43    1.7 2.02
2.4 2.855   3.4 4.045   4.81    5.72    6.805   8.095   9.63    11.46   13.635  16.22   19.3    22.965  27.32   32.505
38.675  46.015];
        expdata = [87.22605 875.2295    1989.45 3172.08 4273.23 5271.7  6163.31 6961.58 7683.185    8350.365
8951.785    9512.44 10054.975   10563.76    11063.475   11489.685   11917.935   12530.3 13649.75    14981.8
16216.2 17500.4 18959.9 20359.35    21705.2 23070.55    24374.5 25565.85    26666.1 27619.25    28400.3 28852
29054.65    28968.85    28591.4 27980.15    27205.4 26320.2 25428.35    24608.75    23847.75    23171.3 22580.45
22025.15    21550   21112.65    20729.8 20377.9 19890.15    19364.75];
        r=1;   %Shear rate
        wt=0.05;

      elseif x==3
        x
        %Data From Will 2 wt%, r=0.1, forward only
        timedata = [0.01    0.02    0.03    0.04    0.05    0.06    0.07    0.08    0.09    0.1 0.11    0.12    0.135   0.165
0.205   0.25    0.31    0.39    0.485   0.6 0.75    0.935   1.165   1.455   1.81    2.255   2.81    3.5 4.365   5.445   6.79
```

```
8.465   10.555  13.165  16.42   20.475  25.53   31.835  39.705  49.515  61.745  77.005  106.585 132.925 165.765
206.725 257.805 321.505 400.935 500.005];
        expdata = [-28.60496667 443.5626667 1193.442667 1972.15 2735.973333 3403.826667 4034.83
4573.966667 5075.016667 5535.47 5957.303333 6345.503333 6907.106667 7933.293333 9092.84 10215.42667
11497.3 12954   14398.53333 15846.76667 17406.13333 18954.4 20514.06667 22172.6 23974.9 25684.93333
27097.3 28527.1 29875.2 30990.9 32182.93333 33062.03333 33815.83333 34364.9 34754.5 34892.8 34841.2
34631.63333 34292.76667 33907.5 33501.1 33124.16667 32801.43333 32584.96667 32437.56667 32364.1
32258.66667 32269.73333 32245.23333 32114.66667];
        r=0.1;   %Shear rate
        wt=0.02;

    elseif x==4
        x
        %Data From Will 5 wt%, r=0.1, forward only
        timedata = [0.01    0.02    0.03    0.04    0.05    0.06    0.07    0.08    0.09    0.1 0.11    0.12    0.135   0.165
0.205   0.25    0.31    0.39    0.485   0.6 0.75    0.935   1.165   1.455   1.81    2.255   2.81    3.5 4.365   5.445   6.79
8.465   10.555  13.165  16.42   20.475  25.53   31.835  39.705  49.515  61.745  77.005  106.585 132.925 165.765
206.725 257.805 321.505 400.935 500.005];
        expdata = [54.64783333  768.4926667 1888.563333 3135.736667 4344.343333 5384    6359.803333
7229.013333 8050.643333 8755.083333 9470.336667 10089.49333 10986.26667 12573.26667 14378.86667
16131.1 18117.26667 20326.36667 22532.76667 24705.8 27045.66667 29492.23333 31878.5 34198.03333
36397.93333 38585.5 40327.9 42087.03333 43897.66667 45157.33333 46242.23333 47230.56667 47865.26667
48410.3 48647.3 48690.86667 48356.6 47634.93333 46584.53333 45316.06667 43983.4 42792.23333 41847.46667
41065.13333 40611.8 40329.6 40256.2 40542.3 40904.36667 40874.16667];
        r=0.1;   %Shear rate
        wt=0.05;
  end



  timedata=timedata*100;
  timedata=round(timedata);
  timedata=timedata/100;

  for i=1:length(sigma)

    for j=1:length(CI)
      [time etatemp]=simulforsigmaCI(r,wt,sigma(i),CI(j),timedata);

      for k=1:length(timedata)
        error(i,j)=error(i,j) + (expdata(k)-etatemp(k))^2;
      end

    end
  end
end

error=error/4;
temp=min(error);
mini=min(temp)
[a b]=find(mini==error);

best_sigma=sigma(a)
best_CI=CI(b)
```

```
surf(CI,sigma,error)
xlabel('CI')
ylabel('sigma')


%Written By Monon Mahboob and Michael Smith
%Edited by William Murch Tim Kremer
%Modeling program for Transient Stress on NanoFiber Compound

function [time etac] = shearsimul(r,massfrac,sigma,CI,tspan)

x=1;

re=33.0; %aspect ratio

chi=1.0*(re^2-1)/(re^2+1); %chi- function of aspect ratio
rf=1750.0; %fiber density
rs=1000.0; %polymer density


etap=[1100.046682   2920.193962 9277.693362 14316.47647 7408.914859]; %Final Parameters after pure polymer
fitting to SAOS and transient
lambda=[0.012287706 0.11728325  0.743829501 2.689760868 15.39198862];
alpha=[0.50413207   0.678743122 0.680016347 0.317377081 0.27975612];




modes=5;


phi=1.0*rs*massfrac/(rf+(rs-rf)*massfrac);  %volume fraction
Ap=1.0*re^2/(3*log(sqrt(1.0*pi/phi)));  %A2, shape factor

orientation=[0.333, 0.0, 0.333];    %assumed initially random

tau_final=zeros(3,modes);

for x=1:modes
   initial=[0 0 0 0 orientation];  %initial conditions
   [time, yo]=ode23tb(@modepolymersub,tspan,initial);
   export = [' mode ' num2str(x) ' calculation done'];
   disp(export)

   L=length(time);

   if x==1
      tau1=zeros(L,modes);
      tau12=zeros(L,modes);
      tau2=zeros(L,modes);
      tau3=zeros(L,modes);
   end
```

```
    tau1(:,x)=yo(:,1);
    tau12(:,x)=yo(:,2);
    tau2(:,x)=yo(:,3);
    tau3(:,x)=yo(:,4);


    tau_final(:,x) = [tau1(L,x) tau12(L,x) tau2(L,x)]';


end

total11=sum(tau1,2);
total12=sum(tau12,2);
total22=sum(tau2,2);
total33=sum(tau3,2);

a11=yo(:,5);
a12=yo(:,6);
a22=yo(:,7);
a33=1-a11-a22;

eta=total12./r;
coef=2.0*eta*phi;

%%%%This section computes the fiber stress from Equation (3)
if r~=0
    disp('r~=0')
    tf12=(coef*Ap*r).*((27.0.*(a11.*a22.*(1-a11-a22)-(a12.^2).*(1-a11-a22))).*(-
1.0/35.0+1.0/7.0.*(a11+a22))+(a12.^2).*(1-27.*(a11.*a22.*(1-a11-a22))+27.0.*(a12.^2).*(1-a11-a22)));
    tf11=(coef*Ap*r).*((27.0.*(a11.*a22.*(1-a11-a22)-(a12.^2).*(1-a11-a22))).*(3.0/7.0.*(a12))+(a11.*a12).*(1-
27*(a11.*a22.*(1-a11-a22))+27.0.*(a12.^2).*(1-a11-a22)));
    tf22=(coef*Ap*r).*((27.0.*(a11.*a22.*(1-a11-a22)-(a12.^2).*(1-a11-a22))).*(3.0/7.0.*(a12))+(a22.*a12).*(1-
27*(a11.*a22.*(1-a11-a22))+27.0.*(a12.^2).*(1-a11-a22)));
    tf33=(coef*Ap*r).*((27.0.*(a11.*a22.*(1-a11-a22)-(a12.^2).*(1-a11-a22))).*(1.0/7.0.*(a12))+((1-a11-
a22).*a12).*(1-27.*(a11.*a22.*(1-a11-a22))+27.0.*(a12.^2).*(1-a11-a22)));
else
    disp('r==0')
    tf12=(2.0*phi.*total12.*Ap).*((27.0*(a11.*a22.*(1-a11-a22)-(a12.^2).*(1-a11-a22))).*(-
1.0/35.0+1.0/7.0*(a11+a22))+(a12.^2).*(1-27*(a11.*a22.*(1-a11-a22))+27.0*(a12.^2).*(1-a11-a22)));
    tf11=(2.0*phi.*total12.*Ap).*((27.0*(a11.*a22.*(1-a11-a22)-(a12.^2).*(1-a11-
a22))).*(3.0/7.0*(a12))+(a11.*a12).*(1-27*(a11.*a22.*(1-a11-a22))+27.0*(a12.^2).*(1-a11-a22)));
    tf22=(2.0*phi.*total12.*Ap).*((27.0*(a11.*a22.*(1-a11-a22)-(a12.^2).*(1-a11-
a22))).*(3.0/7.0*(a12))+(a22.*a12).*(1-27*(a11.*a22.*(1-a11-a22))+27.0*(a12.^2).*(1-a11-a22)));
    tf33=(2.0*phi.*total12.*Ap).*((27.0*(a11.*a22.*(1-a11-a22)-(a12.^2).*(1-a11-a22))).*(1.0/7.0*(a12))+((1-a11-
a22).*a12).*(1-27*(a11.*a22.*(1-a11-a22))+27.0*(a12.^2).*(1-a11-a22)));
end

tauc12=tf12+total12;
tauc11=tf11+total11;
tauc22=tf22+total22;
tauc33=tf33+total33;
etac=tauc12./r;
N1c=tauc11-tauc22;
```

```
N2c=tauc22-tauc33;
aijkl12=((27.0.*(a11.*a22.*(1-a11-a22)-(a12.^2).*(1-a11-a22))).*(-1.0/35.0+1.0/7.0.*(a11+a22))+(a12.^2).*(1-
27.*(a11.*a22.*(1-a11-a22))+27.0.*(a12.^2).*(1-a11-a22)));



disp('finished.')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%
%Function that the ode solvers call to solve Equations (2) and (4) simultaneously
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%
  function dyo = modepolymersub(t,y)

      tp11=y(1);
      tp12=y(2);
      tp22=y(3);
      tp33=y(4);
      a11=y(5);
      a12=y(6);
      a22=y(7);
      a33=1-a11-a22;
      dyo=zeros(7,1);

      dyo(1,1)=-alpha(x)*(tp11^2+tp12^2)/etap(x)-sigma*tp11/lambda(x)+2*r*tp12-3*(1-
sigma)*(tp11*a11+tp12*a12)/lambda(x);
      dyo(2,1)=-alpha(x)*(tp11*tp12+tp12*tp22)/etap(x)-sigma*tp12/lambda(x)+etap(x)*r/lambda(x)+r*tp22-3*(1-
sigma)/2/lambda(x)*(a11*tp12+a12*tp22+a12*tp11+a22*tp12);
      dyo(3,1)=-alpha(x)*(tp12^2+tp22^2)/etap(x)-sigma*tp22/lambda(x)-3*(1-
sigma)/lambda(x)*(tp12*a12+tp22*a22);
      dyo(4,1)=-alpha(x)*(tp33^2)/etap(x)-tp33*sigma/lambda(x)-3*(1-sigma)/lambda(x)*a33*tp33;

      dyo(5,1)=r*a12+2*CI*abs(r)*(1.0-3.0*a11)+chi*r*a12-...
        2*chi*r*(3.0/7.0*a12*(27.0*a11*a22*(1-a11-a22)...
        -27.0*(a12^2)*(1-a11-a22))+(1.0-27.0*a11*a22*(1-a11-a22)+...
        27.0*(a12^2)*(1-a11-a22))*a11*a12);
      dyo(6,1)=1.0/2.0*r*a22-1.0/2.0*r*a11+chi*((1.0/2.0*r*a22+...
        1.0/2.0*r*a11)-(2.0*r)*((27.0*a11*a22*(1-a11-a22)-...
        27.0*(a12^2)*(1-a11-a22))*(-1.0/35.0+1.0/7.0*a11+1.0/7.0*a22)+...
        (1.0-27.0*a11*a22*(1-a11-a22)...
        +27.0*(a12^2)*(1-a11-a22))*a12^2))-6.0*CI*abs(r)*a12;
      dyo(7,1)=-r*a12+2*CI*abs(r)*(1.0-3.0*a22)+chi*r*a12-...
        2*chi*r*(3.0/7.0*a12*(27.0*a11*a22*(1-a11-a22)-...
        27.0*(a12^2)*(1-a11-a22))+(1.0-27.0*a11*a22*(1-a11-a22)+...
        27.0*(a12^2)*(1-a11-a22))*a12*a22);

  end

end
```

Appendix D: MATLAB programs used for calculating error in transient shear viscosity associated with different combinations of σ and C_I in nanocomposite reverse transient shear flows.

```
function error = SigmaCIReverseFit

sigma=0.6:0.02:1;  %Sigmas to test
CI=0.01:0.002:0.1;      %CIs to test

for x=1:12
   x
   if x==1
        %Data From Will 2 wt%, r=0.1, rest time=0, reverse
        timedata = [500.01  500.02  500.03  500.04  500.05  500.06  500.07  500.08  500.09  500.1   500.11  500.12
500.135 500.165 500.205 500.25  500.31  500.39  500.485 500.6   500.75  500.935 501.165 501.455 501.81
502.255 502.81  503.5   504.365 505.445 506.79  508.465 510.555 513.165 516.42  520.475 525.53  531.835
539.705 549.515 561.745 577.005 606.585 632.925 665.765 706.725 757.805 821.505 900.935 1000.005];
        timedata = timedata-timedata(1);
        expdata = [-29422   -28640  -27333  -26013  -24798  -23712  -22605  -21871  -21050  -20303  -19587  -
18978  -18027  -16402  -14468  -12653  -10596  -8178.4 -5827.4 -3399.2 -772.5  1863.37 4482.17 7168.12 9813.94
12493   14848.3 17239.2 19497.3 21630.7 23532.1 25143.1 26648.8 27791.7 28784.8 29511.5 30061.3 30378.3
30588.8 30680.7 30680.4 30666.9 30645   30642.7 30626.6 30746.6 30723.2 30652.1 30765.8 30739.2];
        r=0.1;   %Shear rate
        wt=0.02;
        resttime=0;

        etacmod=zeros(length(timedata),length(sigma),length(CI));
        error=zeros(length(sigma),length(CI));

      elseif x==2
        etacmod=zeros(length(timedata),length(sigma),length(CI));
        %Data From Will 2 wt%, r=0.1, rest time=20, reverse
        expdata = [-1226.1  -785.51 -85.109 679.601 1388.2  2038.59 2608.01 3118.06 3569.61 4009.09 4412.49
4799.2  5321.97 6291.05 7377.11 8456.91 9670.26 10996.8 12357.4 13757.4 15289   16888.1 18504.7 20240.8
21823.8 23212   24774.7 26288.3 27719.7 29078.8 30160.6 31217.1 32114.9 32846.1 33367.6 33811.3 34070.8
34222.6 34285.3 34278.1 34229.4 34167.2 34127.3 34085.3 34064.9 34077.9 34141.4 34042.9 33994.1 33822.2];
        r=0.1;   %Shear rate
        wt=0.02;
        resttime=20;

      elseif x==3
         etacmod=zeros(length(timedata),length(sigma),length(CI));
        %Data From Will 2 wt%, r=0.1, rest time=400, reverse
        expdata = [39.3117  445.593 1122.19 1829.5  2462.48 3075.6  3616.72 4088.46 4547.81 4988.4  5386.87
5740.52 6207.98 7108.95 8151.62 9169.07 10326.5 11657.9 12974   14325.7 15825.5 17416.8 18968.8 20427.2
21846.3 23392.3 24909.5 26336.3 27682.4 28954.3 29983.5 31003.5 31689.8 32321   32750   33001   33128.1
33135.5 33062.5 32931.9 32755.6 32619.7 32511.2 32430.7 32368   32332.9 32233.4 32110.3 32048.6 31856];
        r=0.1;   %Shear rate
        wt=0.02;
        resttime=400;
      elseif x==4
        etacmod=zeros(length(timedata),length(sigma),length(CI));
```

```
        %Data From Will 2 wt%, r=1, rest time=0, reverse
        timedata = [50.01  50.02  50.03  50.04  50.05  50.06  50.07  50.08  50.09  50.1   50.11  50.12  50.13
50.14  50.15  50.16  50.17  50.185 50.215 50.255 50.3   50.355 50.425 50.505 50.6   50.715 50.85  51.01
51.2   51.43  51.7   52.02  52.4   52.855 53.4   54.045 54.81  55.72  56.805 58.095 59.63  61.46  63.635
66.22  69.3   72.965 77.32  82.505 88.675 96.015];
        timedata = timedata-timedata(1);
        expdata = [-15239  -14525 -13370 -12254 -11205 -10290 -9487.7 -8726.8 -8078.3 -7488.3 -6948.7 -
6445.2 -5967.8 -5537.5 -5114.4 -4707.5 -4311  -3775.8 -2785.3 -1612.6 -489.6  703.18  1992.16 3273.77 4486.08
5774.65 7023.2  8207.52 9384.83 10514.7 11535.5 12459  13260.4 13950.3 14511.3 14920.2 15169.5 15268.3
15305.8 15293.3 15230.5 15182.6 15113.8 15044.3 14997.1 14936.4 14913  14806.1 14730.9 14547];
        r=1;  %Shear rate
        wt=0.02;
        resttime=0;
    elseif x==5
     etacmod=zeros(length(timedata),length(sigma),length(CI));
        %Data From Will 2 wt%, r=1, rest time=20, reverse
        expdata = [-80.107  305.417 838.14  1399.41 1912.86 2416.13 2826.59 3235.9  3583.41 3903.97 4186.24
4488.61 4759.96 5031.56 5281.55 5522.38 5736.5  6056.81 6612.53 7291.54 7987.44 8716.92 9526.66 10323.6
11121.2 11903  12664.4 13386.6 14039.9 14612.2 15095.2 15449.1 15691.1 15817.7 15821.2 15709.8 15503.8
15247.1 14981.9 14804  14635.1 14504.4 14317.9 14123  13842.4 13529.6 13244  13176.2 13488.9 13897.3];
        r=1;  %Shear rate
        wt=0.02;
        resttime=20;
    elseif x==6
     etacmod=zeros(length(timedata),length(sigma),length(CI));
        %Data From Will 2 wt%, r=1, rest time=400, reverse
        expdata = [150.635  603.966 1251.89 1934.97 2575.95 3155.9  3684.74 4170.45 4577.09 4979.15 5338.75
5667.57 5981.13 6274.35 6557.39 6821.11 7089.41 7468.58 8159.16 8990.24 9783.55 10670.9 11626.7 12584.7
13481.6 14434.9 15359.6 16208  17003.5 17730.8 18327.7 18756.1 19067.9 19214.7 19224.6 19107.9 18909.3
18645.1 18376.1 18114.6 17891.6 17741.5 17615  17489.4 17362.7 17252.4 17134.7 17075.6 16947.5 16680.3];
        r=1;  %Shear rate
        wt=0.02;
        resttime=400;
    elseif x==7
     etacmod=zeros(length(timedata),length(sigma),length(CI));
        %Data From Will 5 wt%, r=0.1, rest time=0, reverse
        timedata = [500.01  500.02  500.03  500.04  500.05  500.06  500.07  500.08  500.09  500.1   500.11  500.12
500.135 500.165 500.205 500.25  500.31  500.39  500.485 500.6   500.75  500.935 501.165 501.455 501.81
502.255 502.81  503.5   504.365 505.445 506.79  508.465 510.555 513.165 516.42  520.475 525.53  531.835
539.705 549.515 561.745 577.005 606.585 632.925 665.765 706.725 757.805 821.505 900.935 1000.005];
        timedata=timedata-timedata(1);
        expdata = [-44511  -43547 -41853 -40152 -38458 -36973 -35503 -34389 -33072 -32172 -31172 -
30212 -28920 -26727 -24126 -21561 -18596 -15292 -11971 -8552.3 -4788.2 -891.38 3051.97 7115.36 11041.1
14955.1 18859.9 22521.2 26043.9 29189.3 32160.2 34798.3 37007.6 38899.6 40390.5 41512.2 42333.4 42824.7
43110  43222.8 43254.3 43256.8 43260  43294.1 43377.3 43453.1 43554  43635.3 43762.7 43600.4];
        r=0.1;  %Shear rate
        wt=0.05;
        resttime=0;
    elseif x==8
     etacmod=zeros(length(timedata),length(sigma),length(CI));
        %Data From Will 5 wt%, r=0.1, rest time=20, reverse
        expdata = [-1575.4  -1128.5 -445.58 314.253 1016.6  1654.24 2215.37 2762.29 3220.52 3672.55 4066.89
4469.49 4968.89 5894.98 6969.6  8048.37 9276.97 10668.2 12086.8 13594.3 15221.1 16896.4 18592.9 20294
21916.5 23689.8 25359.4 27000.3 28377.5 29761  31058.6 32125.7 33052.5 33797.7 34385.3 34799.3 35085.2
35223.3 35309.5 35323.1 35313.9 35307.5 35334.1 35288.6 35299.8 35297.2 35213.7 35184.4 35445.3 36130.1];
        r=0.1;  %Shear rate
```

```
        wt=0.05;   %[=1 for 2%     =2 for 5%   =3 for 10%  =4 for 0%]
        resttime=20;
    elseif x==9
     etacmod=zeros(length(timedata),length(sigma),length(CI));
        %Data From Will 5 wt%, r=0.1, rest time=400, reverse
        expdata = [27.9889  533.675 1398.34 2312.54 3187.68 3986.05 4693.82 5351.69 5914.99 6452.86 6982.14
7438.94 8077.58 9255.19 10598.4 11882.3 13369.6 15124.6 16777.7 18539.1 20423.1 22336.4 24152.5 26046.5
27921   29933.3 31520.6 33115.9 34712.3 36147.9 37419   38339.2 39223.6 39750.4 40213.8 40470.6 40617.7
40624.4 40620.8 40554.7 40497.2 40481.8 40530.1 40582.4 40669.8 40746.4 40922.7 41040   41349.5 41392];
        r=0.1;   %Shear rate
        wt=0.05;
        resttime=400;
    elseif x==10
        etacmod=zeros(length(timedata),length(sigma),length(CI));
          %Data From Will 5 wt%, r=1, rest time=0, reverse
          timedata=[50.01 50.02   50.03   50.04   50.05   50.06   50.07   50.08   50.09   50.1    50.11   50.12   50.13
50.14   50.15   50.16   50.17   50.185  50.215  50.255  50.3    50.355  50.425  50.505  50.6    50.715  50.85   51.01
51.2    51.43   51.7    52.02   52.4    52.855  53.4    54.045  54.81   55.72   56.805  58.095  59.63   61.46   63.635
66.22   69.3    72.965  77.32   82.505  88.675  96.015];
          timedata = timedata-timedata(1);
          expdata = [-18680   -17884  -16476  -15271  -14034  -12993  -12049  -11155  -10417  -9708   -9066.1 -
8457.8 -7897.3 -7393.8 -6871.9 -6392.3 -5944.2 -5294.7 -4086.5 -2681.8 -1326.9 121.433 1693.95 3276.96 4768.93
6360.23 7900.76 9418.35 10852.5 12142.3 12786.8 13017.7 13478.2 14121.6 15896.9 17581.2 18274.6 18570.7
18619.1 18541.6 18410.8 18222.9 18088.8 18076.6 18097.4 18063.2 18019.3 18060.6 17973.7 17884.4];
          r=1;   %Shear rate
          wt=0.05;
          resttime=0;
    elseif x==11
          etacmod=zeros(length(timedata),length(sigma),length(CI));
            %Data From Will 5 wt%, r=1, rest time=20, reverse
            expdata = [-192.16  270.471 967.685 1684.56 2355.74 2970.66 3533.19 4042.83 4537.7   4962.65
5427.52 5770.33 6144.42 6492.31 6813    7129.61 7397.65 7800.1  8575.99 9506.51 10451.4 11430.4 12516.6
13597.4 14622.8 15674   16697.7 17627   18489.4 19239.6 19897.8 20367   20666.5 20777.1 20749.9 20603
20353.4 19954.6 19521.8 19130.7 18819.6 18487.7 18166.7 17888.2 17724.7 17580.8 17305.3 16775   16199.1
15827.4];
            r=1;   %Shear rate
            wt=0.05;
            resttime=20;
    elseif x==12
           etacmod=zeros(length(timedata),length(sigma),length(CI));
             %Data From Will 5 wt%, r=1, rest time=400, reverse
             expdata = [224.512  696.655 1364.99 2045.32 2683.89 3258.89 3761.66 4219.15 4637.74 5023.27
5391    5748.56 6051.95 6373.39 6674.75 6946.86 7224.56 7609.71 8299.53 9116.11 9944.7  10788.7 11749.4
12690.4 13577.3 14516.7 15420.1 16280.7 17043.8 17772   18348.7 18782.2 19099.4 19268.1 19288.5 19191.3
19015.6 18782.8 18535.8 18303.2 18114.2 17973.1 17855.4 17751.6 17661.3 17585.9 17516.2 17359.5 17168.6
16968.6];
             r=1;   %Shear rate
             wt=0.05;
             resttime=400;

  end

  timedata=timedata*100;
  timedata=round(timedata);
  timedata=timedata/100;
```

```
   for i=1:length(sigma)
      for j=1:length(CI)
         [time etatemp]=SimulReverse(r,wt,sigma(i),CI(j),resttime,timedata);


         for k=1:length(timedata)
            error(i,j)=error(i,j) + (expdata(k)-etatemp(k))^2;
         end
      end
   end
end


x

error=error/x
sigma
CI

temp=min(error);
mini=min(temp)
[a b]=find(mini==error);

best_sigma=sigma(a)
best_CI=CI(b)

surf(CI,sigma,error)
xlabel('CI')
ylabel('sigma')


end



%Written By Monon Mahboob and Michael Smith
%Edited by William Murch Tim Kremer
%Modeling program for Transient Stress on NanoFiber Compounds

function [timex etacr] = SimulReverse(r1,massfrac,sigma,CI,resttime,tspanr)

x=1;

re=33.0; %aspect ratio

chi=1.0*(re^2-1)/(re^2+1); %another form of aspect ratio
rf=1750.0; %fiber density
rs=1000.0; %polymer density
```

```
etap=[1100.046682   2920.193962 9277.693362 14316.47647 7408.914859]; %Final Parameters after pure polymer fitting to SAOS and transient
lambda=[0.012287706 0.11728325  0.743829501 2.689760868 15.39198862];
alpha=[0.50413207   0.678743122 0.680016347 0.317377081 0.27975612];

r=r1;
modes=5;

tspanf=tspanr-tspanr(1);

phi=1.0*rs*massfrac/(rf+(rs-rf)*massfrac);  %volume fraction
Ap=1.0*re^2/(3*log(sqrt(1.0*pi/phi)));  %A2, shape factor

orientation=[0.333, 0.0, 0.333];

tau_finalf=zeros(4,modes);

for x=1:modes
   initial=[0 0 0 0 orientation];
   [time1, yo]=ode23tb(@modepolymersub,tspanf,initial);
   export = [' mode ' num2str(x) ' forward calculation done'];

     L=length(time1);

   if x==1
      tau1=zeros(L,modes);
      tau12=zeros(L,modes);
      tau2=zeros(L,modes);
      tau3=zeros(L,modes);
   end

   tau1(:,x)=yo(:,1);
   tau12(:,x)=yo(:,2);
   tau2(:,x)=yo(:,3);
   tau3(:,x)=yo(:,4);


   tau_finalf(:,x) = [tau1(L,x) tau12(L,x) tau2(L,x) tau3(L,x)]';

end

total11f=sum(tau1,2);
total12f=sum(tau12,2);
total22f=sum(tau2,2);
total33f=sum(tau3,2);

a11f=yo(:,5);
a12f=yo(:,6);
a22f=yo(:,7);
a33f=1-a11f-a22f;

disp('final values for T11p T12p T22p T33p Forward')
T_final= [total11f(L) total12f(L) total22f(L) total33f(L)];
```

```
disp(T_final)

disp('mode forward stress finals T11 T12 T22')
disp(tau_finalf)

disp('forward final values for a11 a12 a22 a33 =')
a_final = [a11f(L) a12f(L) a22f(L)]';
disp(a_final)

etaf=total12f./r;
coef=2.0*etaf*phi;

%%%%This section computes the fiber stress from Equation (3)
if r~=0
   disp('r~=0')
   tf12f=(coef*Ap*r).*((27.0.*(a11f.*a22f.*(1-a11f-a22f)-(a12f.^2).*(1-a11f-a22f))).*(-
1.0/35.0+1.0/7.0.*(a11f+a22f))+(a12f.^2).*(1-27.*(a11f.*a22f.*(1-a11f-a22f))+27.0.*(a12f.^2).*(1-a11f-a22f)));
   tf11f=(coef*Ap*r).*((27.0.*(a11f.*a22f.*(1-a11f-a22f)-(a12f.^2).*(1-a11f-
a22f))).*(3.0/7.0.*(a12f))+(a11f.*a12f).*(1-27*(a11f.*a22f.*(1-a11f-a22f))+27.0.*(a12f.^2).*(1-a11f-a22f)));
   tf22f=(coef*Ap*r).*((27.0.*(a11f.*a22f.*(1-a11f-a22f)-(a12f.^2).*(1-a11f-
a22f))).*(3.0/7.0.*(a12f))+(a22f.*a12f).*(1-27*(a11f.*a22f.*(1-a11f-a22f))+27.0.*(a12f.^2).*(1-a11f-a22f)));
   tf33f=(coef*Ap*r).*((27.0.*(a11f.*a22f.*(1-a11f-a22f)-(a12f.^2).*(1-a11f-a22f))).*(1.0/7.0.*(a12f))+((1-a11f-
a22f).*a12f).*(1-27.*(a11f.*a22f.*(1-a11f-a22f))+27.0.*(a12f.^2).*(1-a11f-a22f)));
else
   disp('r==0')
   tf12f=(2.0*phi.*total12.*Ap).*((27.0*(a11f.*a22f.*(1-a11f-a22f)-(a12f.^2).*(1-a11f-a22f))).*(-
1.0/35.0+1.0/7.0*(a11f+a22f))+(a12f.^2).*(1-27*(a11f.*a22f.*(1-a11f-a22f))+27.0*(a12f.^2).*(1-a11f-a22f)));
   tf11f=(2.0*phi.*total12.*Ap).*((27.0*(a11f.*a22f.*(1-a11f-a22f)-(a12f.^2).*(1-a11f-
a22f))).*(3.0/7.0*(a12f))+(a11f.*a12f).*(1-27*(a11f.*a22f.*(1-a11f-a22f))+27.0*(a12f.^2).*(1-a11f-a22f)));
   tf22f=(2.0*phi.*total12.*Ap).*((27.0*(a11f.*a22f.*(1-a11f-a22f)-(a12f.^2).*(1-a11f-
a22f))).*(3.0/7.0*(a12f))+(a22f.*a12f).*(1-27*(a11f.*a22f.*(1-a11f-a22f))+27.0*(a12f.^2).*(1-a11f-a22f)));
   tf33f=(2.0*phi.*total12.*Ap).*((27.0*(a11f.*a22f.*(1-a11f-a22f)-(a12f.^2).*(1-a11f-
a22f))).*(1.0/7.0*(a12f))+((1-a11f-a22f).*a12f).*(1-27*(a11f.*a22f.*(1-a11f-a22f))+27.0*(a12f.^2).*(1-a11f-
a22f)));
end

tauc12f=tf12f+total12f;
tauc11f=tf11f+total11f;
tauc22f=tf22f+total22f;
tauc33f=tf33f+total33f;
etacf=tauc12f./r;

%-------------------------------------------------------------------------
%   RELAXATION PERIOD
%-------------------------------------------------------------------------
if resttime~=0
tau_finalm=zeros(4,modes);
r=0;

for x=1:modes
   initial2=[tau_finalf(:,x); a_final];
   tspan2=time1(end):resttime+time1(end);
   [time2, yo]=ode23tb(@modepolymersub,tspan2,initial2);
   export = [' mode ' num2str(x) ' relaxation calculation done'];
```

```
    disp(export)

    L=length(time2);

    if x==1
        tau1=zeros(L,modes);
        tau12=zeros(L,modes);
        tau2=zeros(L,modes);
        tau3=zeros(L,modes);
    end

    tau1(:,x)=yo(:,1);
    tau12(:,x)=yo(:,2);
    tau2(:,x)=yo(:,3);
    tau3(:,x)=yo(:,4);

    tau_finalm(:,x) = [tau1(L,x) tau12(L,x) tau2(L,x) tau3(L,x)]';

end

total11m=sum(tau1,2);
total12m=sum(tau12,2);
total22m=sum(tau2,2);
total33m=sum(tau3,2);

a11m=yo(:,5);
a12m=yo(:,6);
a22m=yo(:,7);
a33m=1-a11m-a22m;

disp('final relaxation values for T11p T12p T22p T33p')
T_final= [total11m(L) total12m(L) total22m(L) total33m(L)];
disp(T_final)

disp('mode relaxation stress finals T11 T12 T22')
disp(tau_finalm)

disp('relaxation final values for a11 a12 a22 a33 =')
a_final = [a11m(L) a12m(L) a22m(L)]';
disp(a_final)

%%%%This section computes the fiber stress from Equation (3)
if r~=0
    error('r~=0')
else
    disp('r==0')
    tf12m=(2.0*phi.*total12m.*Ap).*((27.0*(a11m.*a22m.*(1-a11m-a22m)-(a12m.^2).*(1-a11m-a22m))).*(-
1.0/35.0+1.0/7.0*(a11m+a22m))+(a12m.^2).*(1-27*(a11m.*a22m.*(1-a11m-a22m))+27.0*(a12m.^2).*(1-a11m-
a22m)));
    tf11m=(2.0*phi.*total12m.*Ap).*((27.0*(a11m.*a22m.*(1-a11m-a22m)-(a12m.^2).*(1-a11m-
a22m))).*(3.0/7.0*(a12m))+(a11m.*a12m).*(1-27*(a11m.*a22m.*(1-a11m-a22m))+27.0*(a12m.^2).*(1-a11m-
a22m)));
```

```
   tf22m=(2.0*phi.*total12m.*Ap).*((27.0*(a11m.*a22m.*(1-a11m-a22m)-(a12m.^2).*(1-a11m-
a22m))).*(3.0/7.0*(a12m))+(a22m.*a12m).*(1-27*(a11m.*a22m.*(1-a11m-a22m))+27.0*(a12m.^2).*(1-a11m-
a22m)));
   tf33m=(2.0*phi.*total12m.*Ap).*((27.0*(a11m.*a22m.*(1-a11m-a22m)-(a12m.^2).*(1-a11m-
a22m))).*(1.0/7.0*(a12m))+((1-a11m-a22m).*a12m).*(1-27*(a11m.*a22m.*(1-a11m-a22m))+27.0*(a12m.^2).*(1-
a11m-a22m)));
end

tauc12m=tf12m+total12m;
tauc11m=tf11m+total11m;
tauc22m=tf22m+total22m;
tauc33m=tf33m+total33m;
etacm=tauc12m./r;
end

%-------------------------------------------------------------------------
%   REVERSE  DIRECTION
%-------------------------------------------------------------------------
tau_finalr=zeros(4,modes);
r=-r1;

for x=1:modes
   if resttime~=0
      initialr=[tau_finalm(:,x); a_final];
   else
      initialr=[tau_finalf(:,x); a_final];
   end

   [timer, yo]=ode23tb(@modepolymersub,tspanr,initialr);
   export = [' mode ' num2str(x) ' reverse calculation done'];
   disp(export)

   L=length(timer);

   if x==1
      tau1=zeros(L,modes);
      tau12=zeros(L,modes);
      tau2=zeros(L,modes);
      tau3=zeros(L,modes);
   end

   tau1(:,x)=yo(:,1);
   tau12(:,x)=yo(:,2);
   tau2(:,x)=yo(:,3);
   tau3(:,x)=yo(:,4);



   tau_finalr(:,x) = [tau1(L,x) tau12(L,x) tau2(L,x) tau3(L,x)]';

end

total11r=sum(tau1,2);
total12r=sum(tau12,2);
total22r=sum(tau2,2);
```

```
total33r=sum(tau3,2);

a11r=yo(:,5);
a12r=yo(:,6);
a22r=yo(:,7);
a33r=1-a11r-a22r;

disp('final reverse values for T11p T12p T22p T33p')
T_final= [total11r(L) total12r(L) total22r(L) total33r(L)];
disp(T_final)

disp('mode reverse stress finals T11 T12 T22')
disp(tau_finalr)

disp('reverse final values for a11 a12 a22 a33 =')
a_final = [a11r(L) a12r(L) a22r(L) a33r(L)];
disp(a_final)

etar=total12r./r;
coef=2.0*etar*phi;

%%%%This section computes the fiber stress from Equation (3)
if r~=0
   disp('r~=0')
   tf12r=(coef*Ap*r).*((27.0.*(a11r.*a22r.*(1-a11r-a22r)-(a12r.^2).*(1-a11r-a22r))).*(-
1.0/35.0+1.0/7.0.*(a11r+a22r))+(a12r.^2).*(1-27.*(a11r.*a22r.*(1-a11r-a22r))+27.0.*(a12r.^2).*(1-a11r-a22r)));
   tf11r=(coef*Ap*r).*((27.0.*(a11r.*a22r.*(1-a11r-a22r)-(a12r.^2).*(1-a11r-
a22r))).*(3.0/7.0.*(a12r))+(a11r.*a12r).*(1-27*(a11r.*a22r.*(1-a11r-a22r))+27.0.*(a12r.^2).*(1-a11r-a22r)));
   tf22r=(coef*Ap*r).*((27.0.*(a11r.*a22r.*(1-a11r-a22r)-(a12r.^2).*(1-a11r-
a22r))).*(3.0/7.0.*(a12r))+(a22r.*a12r).*(1-27*(a11r.*a22r.*(1-a11r-a22r))+27.0.*(a12r.^2).*(1-a11r-a22r)));
   tf33r=(coef*Ap*r).*((27.0.*(a11r.*a22r.*(1-a11r-a22r)-(a12r.^2).*(1-a11r-a22r))).*(1.0/7.0.*(a12r))+((1-a11r-
a22r).*a12r).*(1-27.*(a11r.*a22r.*(1-a11r-a22r))+27.0.*(a12r.^2).*(1-a11r-a22r)));
else
   error('r==0')

end

tauc12r=tf12r+total12r;
tauc11r=tf11r+total11r;
tauc22r=tf22r+total22r;
tauc33r=tf33r+total33r;
etacr=tauc12r./r;

%-------------------------------------------------------------------------
%   COMBINE AND GRAPH
%-------------------------------------------------------------------------

if resttime~=0
   tauc12=[tauc12f;tauc12m;tauc12r];
   tauc11=[tauc11f;tauc11m;tauc11r];
   tauc22=[tauc22f;tauc22m;tauc22r];
   etac=[etacf;etacm;etacr];
   time=[time1;time2;timer];
end
```

```
timex=timer-time1(end)-resttime;

graph=0;
if graph==1
   figure;
   semilogx(time,etac)

   figure(2);
   semilogx(time1,tauc12f,time1,tauc11f,time1,tauc22f)
   legend('tauc12','tauc11','tauc22')
   title('Forward')

   figure(3);
   semilogx(time2-time1(L1),tauc12r,time2-time1(L1),tauc11r,time2-time1(L1),tauc22r)
   legend('tauc12','tauc11','tauc22')
   title('Reverse')

   figure(4);
   semilogx(time1,total12f,time1,total11f,time1,total22f)
   legend('tauc12','tauc11','tauc22')
   title('Forward')
end


   function dyo = modepolymersub(t,y)

      tp11=y(1);
      tp12=y(2);
      tp22=y(3);
      tp33=y(4);
      a11=y(5);
      a12=y(6);
      a22=y(7);
      a33=1-a11-a22;
      dyo=zeros(7,1);

      dyo(1,1)=-alpha(x)*(tp11^2+tp12^2)/etap(x)-sigma*tp11/lambda(x)+2*r*tp12-3*(1-
sigma)*(tp11*a11+tp12*a12)/lambda(x);
      dyo(2,1)=-alpha(x)*(tp11*tp12+tp12*tp22)/etap(x)-sigma*tp12/lambda(x)+etap(x)*r/lambda(x)+r*tp22-3*(1-
sigma)/2/lambda(x)*(a11*tp12+a12*tp22+a12*tp11+a22*tp12);
      dyo(3,1)=-alpha(x)*(tp12^2+tp22^2)/etap(x)-sigma*tp22/lambda(x)-3*(1-
sigma)/lambda(x)*(tp12*a12+tp22*a22);
      dyo(4,1)=-alpha(x)*(tp33^2)/etap(x)-tp33*sigma/lambda(x)-3*(1-sigma)/lambda(x)*a33*tp33;

      dyo(5,1)=r*a12+2*CI*abs(r)*(1.0-3.0*a11)+chi*r*a12-...
         2*chi*r*(3.0/7.0*a12*(27.0*a11*a22*(1-a11-a22)...
         -27.0*(a12^2)*(1-a11-a22))+(1.0-27.0*a11*a22*(1-a11-a22)+...
         27.0*(a12^2)*(1-a11-a22))*a11*a12);
      dyo(6,1)=1.0/2.0*r*a22-1.0/2.0*r*a11+chi*((1.0/2.0*r*a22+...
         1.0/2.0*r*a11)-(2.0*r)*((27.0*a11*a22*(1-a11-a22)-...
         27.0*(a12^2)*(1-a11-a22))*(-1.0/35.0+1.0/7.0*a11+1.0/7.0*a22)+...
         (1.0-27.0*a11*a22*(1-a11-a22)...
         +27.0*(a12^2)*(1-a11-a22))*a12^2))-6.0*CI*abs(r)*a12;
```

```
        dyo(7,1)=-r*a12+2*CI*abs(r)*(1.0-3.0*a22)+chi*r*a12-...
            2*chi*r*(3.0/7.0*a12*(27.0*a11*a22*(1-a11-a22)-...
            27.0*(a12^2)*(1-a11-a22))+(1.0-27.0*a11*a22*(1-a11-a22)+...
            27.0*(a12^2)*(1-a11-a22))*a12*a22);

    end

end
```

Appendix E: Model predictions vs. experimental viscosity data for nanocomposites in forward and reverse flows with optimized σ and C_I values.

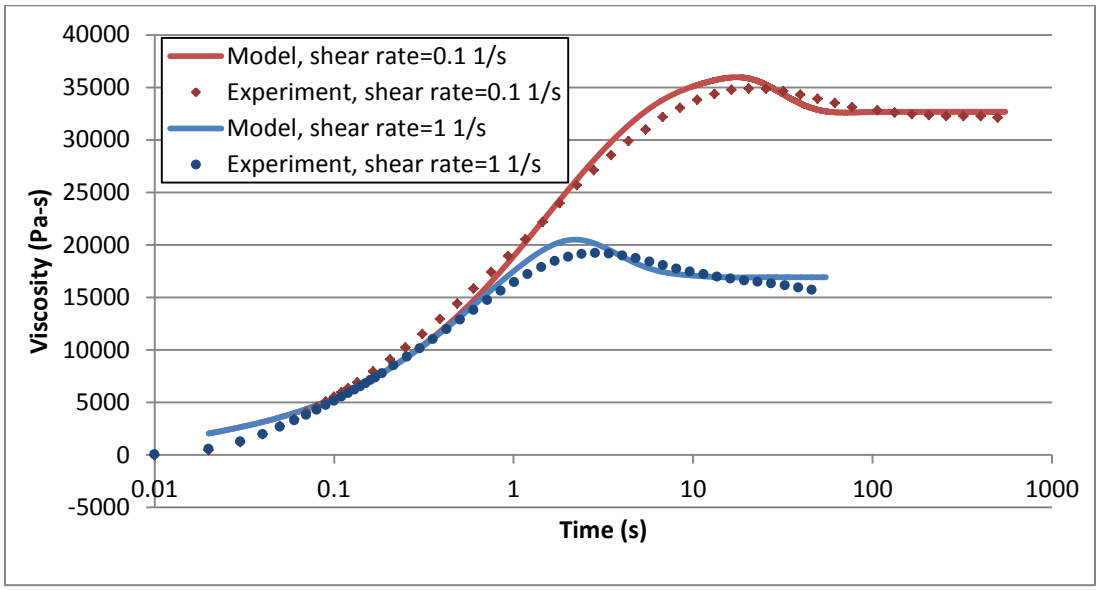


Figure 23: Model predictions and experimental viscosity in a forward shear flow of a 2wt% CNF composite.

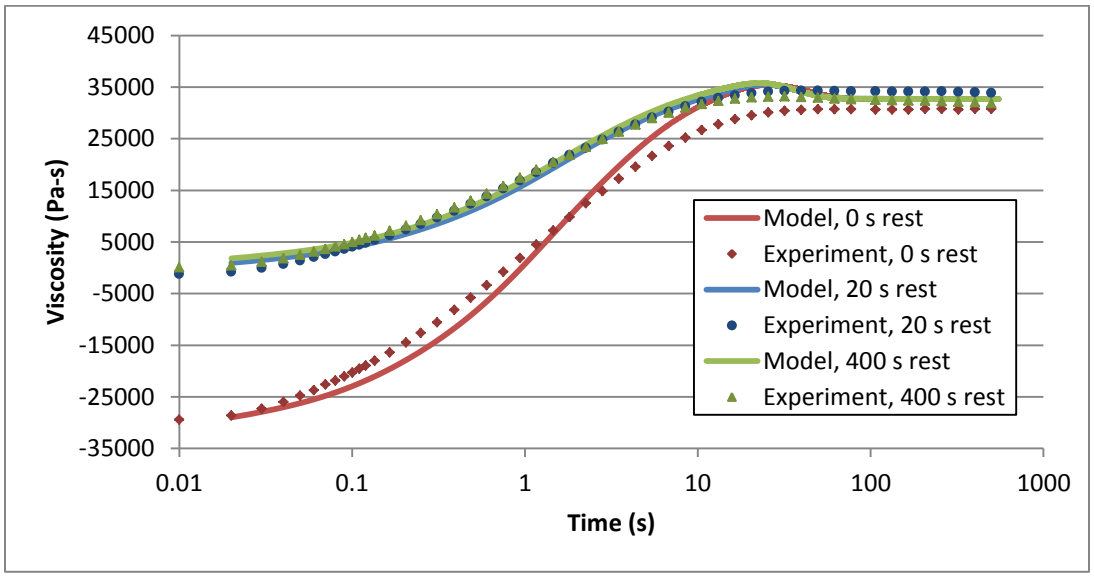


Figure 24: Model predictions and experimental viscosity for a 2wt% nanocomposite in a reverse flow at a shear rate of 0.1 s^{-1}.

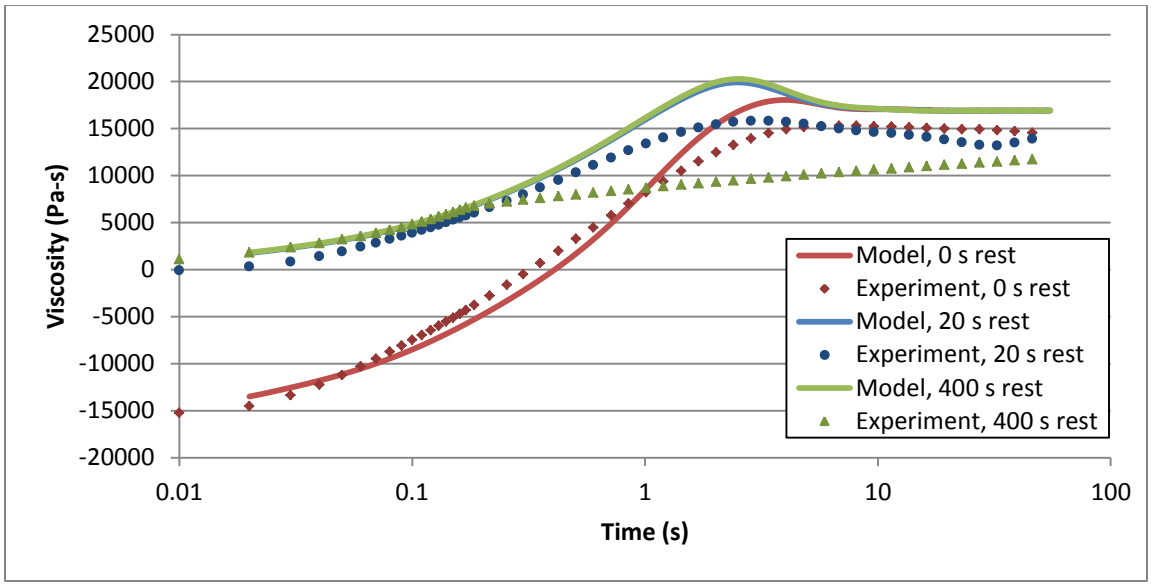


Figure 25: Model predictions and experimental viscosity for a 2wt% nanocomposite in a reverse flow at a shear rate of 1 s^{-1}.

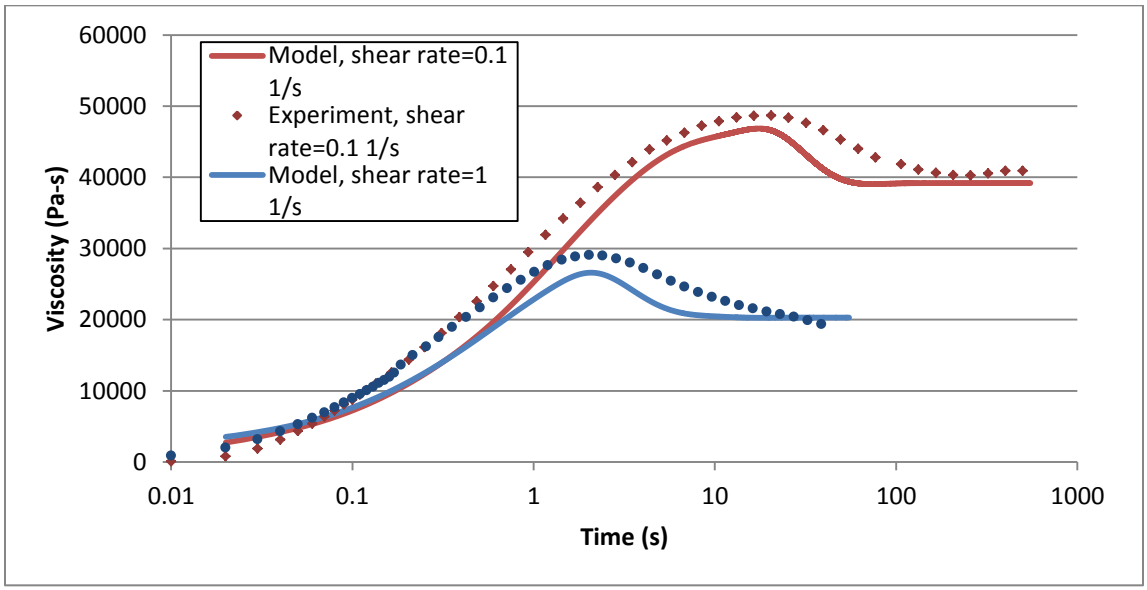


Figure 26: Model predictions and experimental viscosity in a forward shear flow of a 5wt% CNF composite.

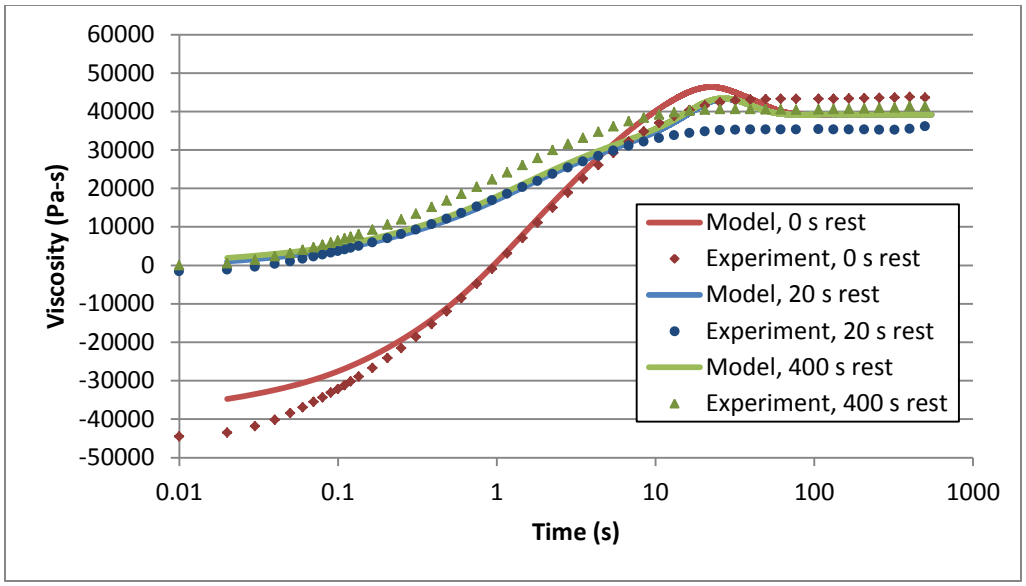


Figure 27: Model predictions and experimental viscosity for a 5wt% nanocomposite in a reverse flow at a shear rate of 0.1 s^{-1}.

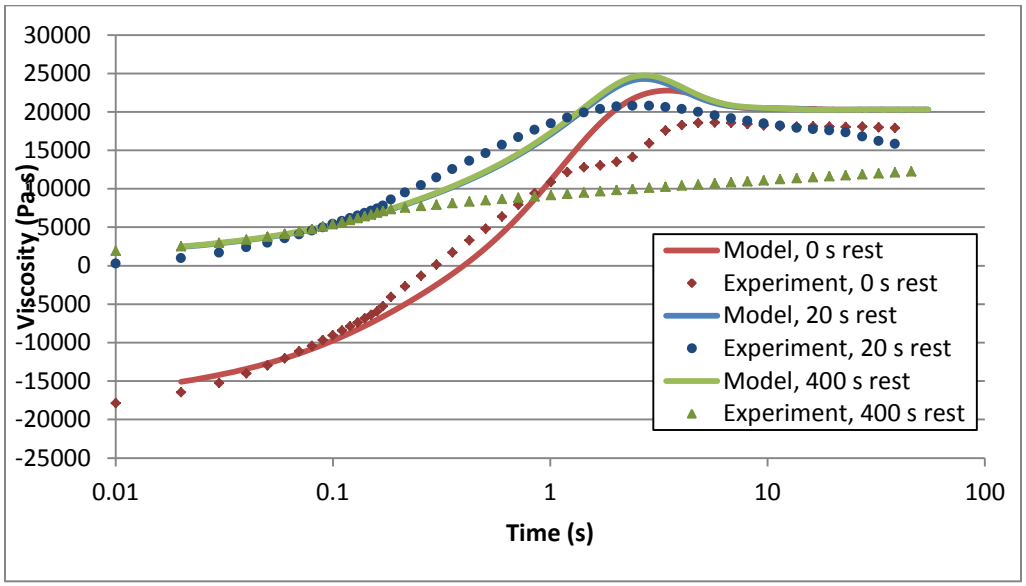


Figure 28: Model predictions and experimental viscosity for a 5wt% nanocomposite in a reverse flow at a shear rate of 1 s^{-1}.

Appendix F: Normal stress model predictions vs. experimental data in forward transient shear flows.

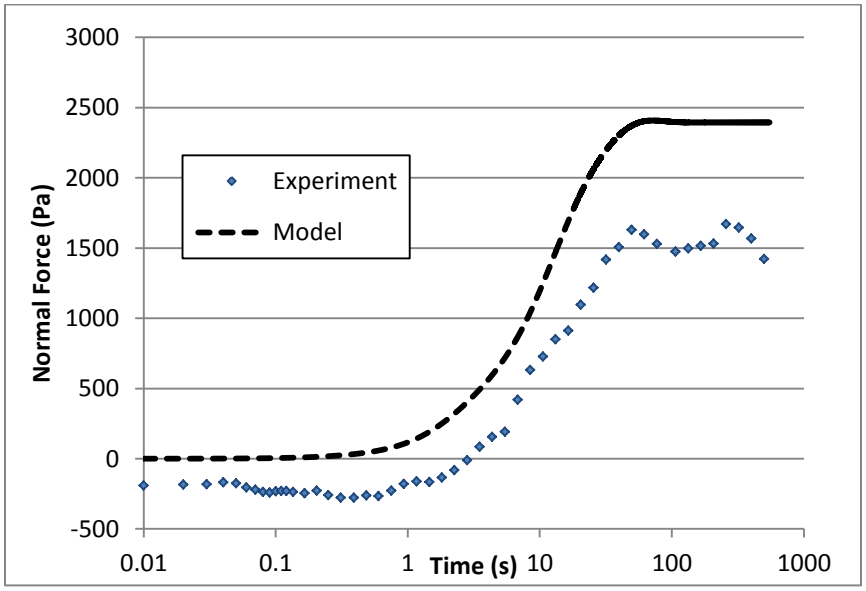


Figure 29: Normal stress model predictions vs. experimental data for the pure polymer at a shear rate of 0.1 s^{-1}.

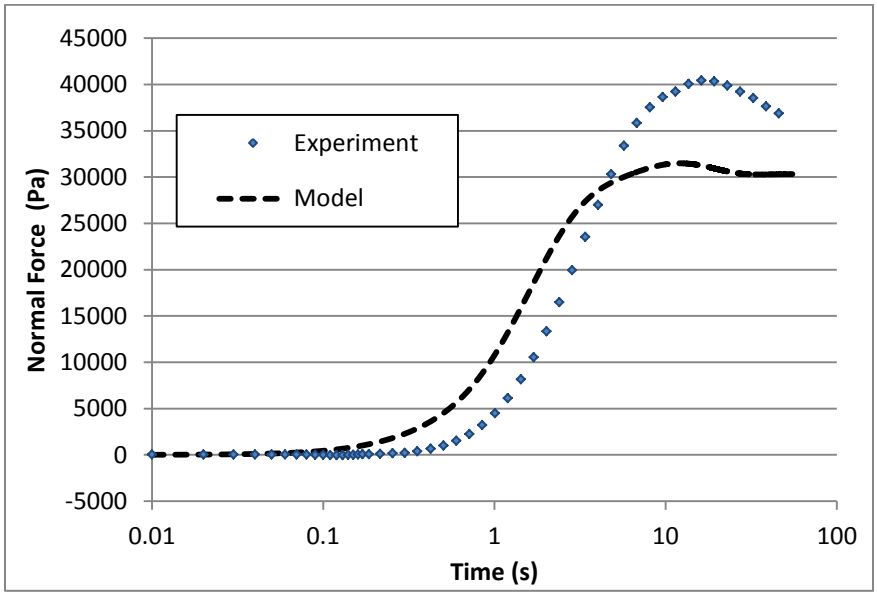


Figure 30: Normal stress model predictions vs. experimental data for the pure polymer at a shear rate of 1 s^{-1}.

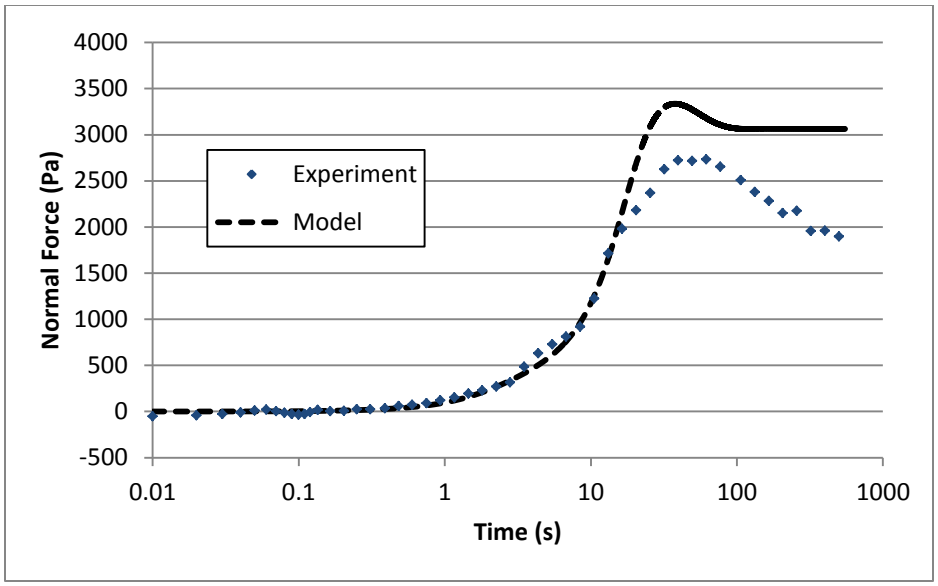


Figure 31: Normal stress model predictions vs. experimental data for a 2wt% CNF nanocomposite at a shear rate of 0.1 s^{-1}.

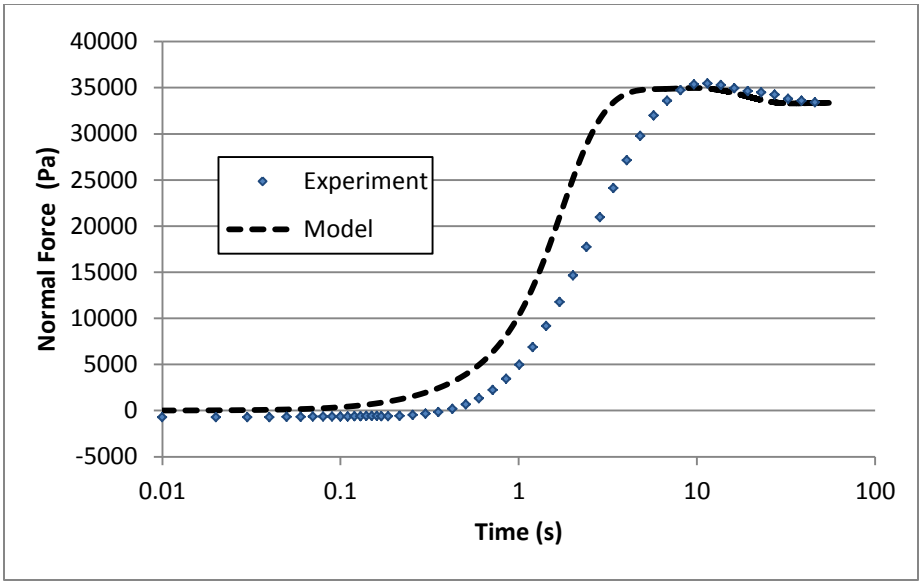


Figure 32: Normal stress model predictions vs. experimental data for a 2wt% CNF nanocomposite at a shear rate of 1 s^{-1}.

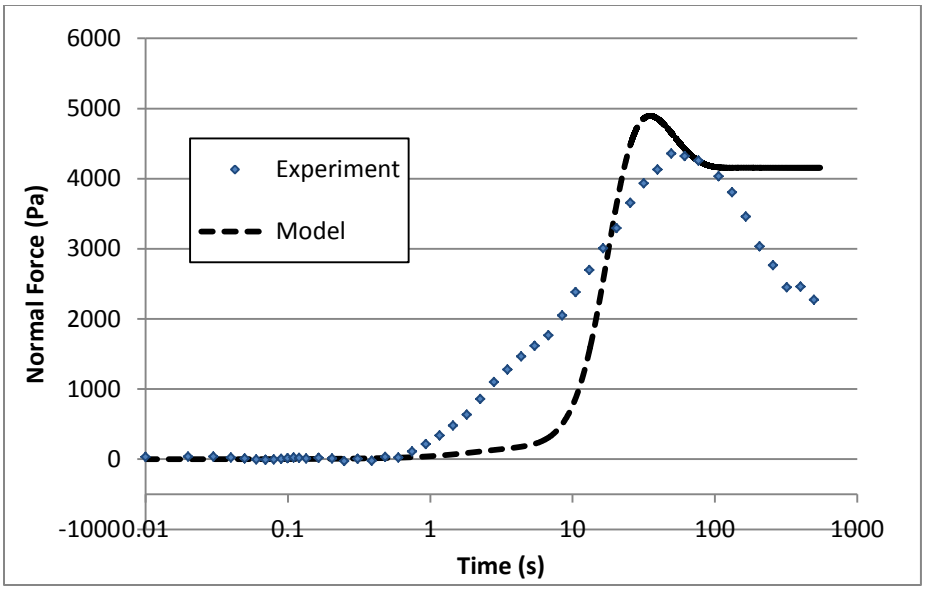


Figure 33: Normal stress model predictions vs. experimental data for a 5wt% CNF nanocomposite at a shear rate of 0.1 s^{-1}.

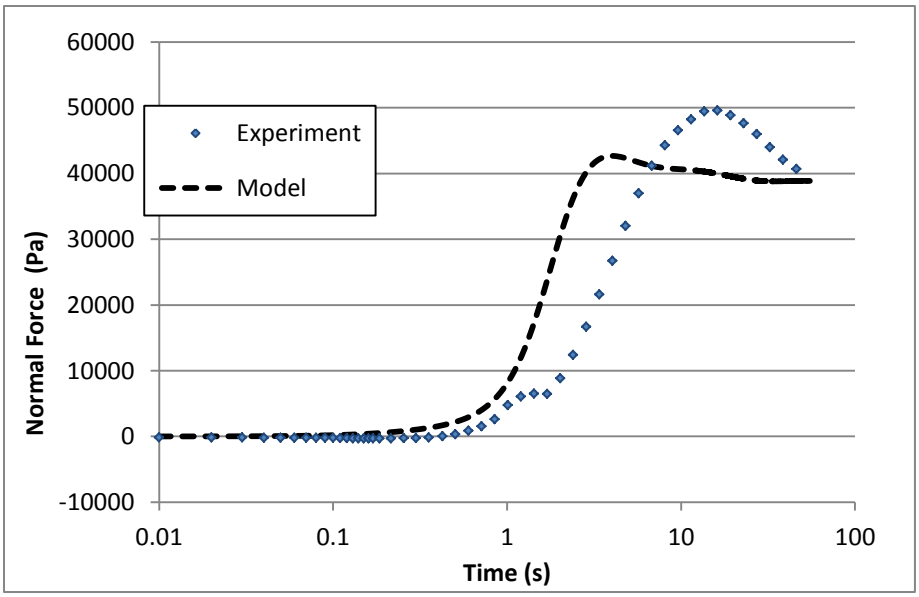


Figure 34: Normal stress model predictions vs. experimental data for a 5wt% CNF nanocomposite at a shear rate of 1 s^{-1}.

Appendix G: MATLAB program for modeling stress wave from small amplitude oscillatory shear (SAOS) flows

```
function [export] = SAOSModeling
%This programmodels a small amplitude oscillatory shear (SAOS) flow and
%graphs the resulting stress wave

x=1;
CI=0.036;   %particle-particle interaction parameter
sigma=0.84;  %particle-polymer interaction parameter
r0=0.5; %percent

c=1; %wt% CNFs, 1=2%, 2=5%, 3=10%, 4=0%

freq=[0.01585   0.02512 0.03981 0.0631  0.1 0.15849 0.25119 0.39811 0.63096 1   1.58489 2.51189 3.98107
6.30957 10  15.8489 25.1189 39.8107 63.0957 100];

if c==4
   freq=[0.01585   0.02512 0.03981 0.0631  0.1 0.15849 0.25119 0.39811 0.63096 1   1.58489 2.51189 3.98107
6.30957 10  15.8489 25.1189 39.8107 63.0957 100];
   Gp=[35.06125    118.77715   218.069 403.069 731.4956667 1295.72 2281.803333 3723.183333 5670.693333
8656.983333 12385.93333 17431.8 23216.63333 30205.2 38639.13333 47962.26667 58520.46667 70267.56667
82679.76667 96278.46667];
   Gdp=[532.1383333    799.4886667 1259.376667 1859.03 2733.77 3907.66 5450.736667 7393.43 9767.963333
12548.53333 15681.63333 19139.5 22729.5 26576.8 30379.83333 34434.86667 38387.86667 42793.26667 47297.4
52348.03333];
elseif c==1
   Gp=[55.6626 118.492 254.715 503.824 908.168 1605.89 2780.195    4406.575    6593.61 10148.9 14452.5
20191.5 26905.4 34823.55    44354.4 55135.45    67152.35    80926.2 95075.9 110655];
   Gdp=[618.6515   916.983 1479.19 2195.635    3208.705    4568.785    6359.81 8571.75 11351.8 14440.1
18048.75    21982.25    26022.35    30520.1 35164.85    39549.5 44375.45    49176.75    54481   60832];
end


ref=10;    %index for frequency
w=freq(ref);    %index for frequency

re=33.0; %aspect ratio

chi=1.0*(re^2-1)/(re^2+1); %another form of aspect ratio
rf=1750.0; %fiber density
rs=1000.0; %polymer density


etap=[1100.046682   2920.193962 9277.693362 14316.47647 7408.914859]; %Final Parameters after pure polymer
fitting to SAOS and transient
lambda=[0.012287706 0.11728325  0.743829501 2.689760868 15.39198862];
alpha=[0.50413207   0.678743122 0.680016347 0.317377081 0.27975612];

tspan=0:1/10/w:10*pi/w;

mass=[2.0/100.0, 5.0/100.0, 10.0/100.0, 0.0];
```

```
modes=5;
%tspan=0:0.01:shear_time;


phi=1.0*rs*mass(c)/(rf+(rs-rf)*mass(c));  %volume fraction
Ap=1.0*re^2/(3*log(sqrt(1.0*pi/phi)));  %A2, shape factor

orientation=[0.333, 0.0, 0.333];

tau_final=zeros(3,modes);


while x<modes+1
   initial=[0 0 0 0 orientation];
   [time, yo]=ode45(@modepolymersub,tspan,initial);
   export = [' mode ' num2str(x) ' calculation done'];
   disp(export)

   L=length(time);

   if x==1
      tau1=zeros(L,modes);
      tau12=zeros(L,modes);
      tau2=zeros(L,modes);
      tau3=zeros(L,modes);
   end

   tau1(:,x)=yo(:,1);
   tau12(:,x)=yo(:,2);
   tau2(:,x)=yo(:,3);
   tau3(:,x)=yo(:,4);

   r=r0*w*cos(w*time);

   subplot(1,3,1)
   plot(time,tau12)
   title('Individual Modes')

   hold on

   tau_final(:,x) = [tau1(L) tau12(L) tau2(L)]';


x=x+1;


end

total11=sum(tau1,2);
total12=sum(tau12,2);
total22=sum(tau2,2);
```

```
total33=sum(tau3,2);

a11=yo(:,5);
a12=yo(:,6);
a22=yo(:,7);
a33=1-a11-a22;

eta=total12./r;
coef=2.0*eta*phi;

if r~=0
   disp('r~=0')
   tf12=(coef.*Ap.*r).*((27.0.*(a11.*a22.*(1-a11-a22)-(a12.^2).*(1-a11-a22))).*(-
1.0/35.0+1.0/7.0.*(a11+a22))+(a12.^2).*(1-27.*(a11.*a22.*(1-a11-a22))+27.0.*(a12.^2).*(1-a11-a22)));
   tf11=(coef.*Ap.*r).*((27.0.*(a11.*a22.*(1-a11-a22)-(a12.^2).*(1-a11-a22))).*(3.0/7.0.*(a12))+(a11.*a12).*(1-
27*(a11.*a22.*(1-a11-a22))+27.0.*(a12.^2).*(1-a11-a22)));
   tf22=(coef.*Ap.*r).*((27.0.*(a11.*a22.*(1-a11-a22)-(a12.^2).*(1-a11-a22))).*(3.0/7.0.*(a12))+(a22.*a12).*(1-
27*(a11.*a22.*(1-a11-a22))+27.0.*(a12.^2).*(1-a11-a22)));
   tf33=(coef.*Ap.*r).*((27.0.*(a11.*a22.*(1-a11-a22)-(a12.^2).*(1-a11-a22))).*(1.0/7.0.*(a12))+((1-a11-
a22).*a12).*(1-27.*(a11.*a22.*(1-a11-a22))+27.0.*(a12.^2).*(1-a11-a22)));
else
   disp('r==0')
   tf12=(2.0*phi.*total12.*Ap).*((27.0*(a11.*a22.*(1-a11-a22)-(a12.^2).*(1-a11-a22))).*(-
1.0/35.0+1.0/7.0*(a11+a22))+(a12.^2).*(1-27*(a11.*a22.*(1-a11-a22))+27.0*(a12.^2).*(1-a11-a22)));
   tf11=(2.0*phi.*total12.*Ap).*((27.0*(a11.*a22.*(1-a11-a22)-(a12.^2).*(1-a11-
a22))).*(3.0/7.0*(a12))+(a11.*a12).*(1-27*(a11.*a22.*(1-a11-a22))+27.0*(a12.^2).*(1-a11-a22)));
   tf22=(2.0*phi.*total12.*Ap).*((27.0*(a11.*a22.*(1-a11-a22)-(a12.^2).*(1-a11-
a22))).*(3.0/7.0*(a12))+(a22.*a12).*(1-27*(a11.*a22.*(1-a11-a22))+27.0*(a12.^2).*(1-a11-a22)));
   tf33=(2.0*phi.*total12.*Ap).*((27.0*(a11.*a22.*(1-a11-a22)-(a12.^2).*(1-a11-a22))).*(1.0/7.0*(a12))+((1-a11-
a22).*a12).*(1-27*(a11.*a22.*(1-a11-a22))+27.0*(a12.^2).*(1-a11-a22)));
end

tauc12=tf12+total12;
tauc11=tf11+total11;
tauc22=tf22+total22;
tauc33=tf33+total33;
etac=tauc12./r;
N1c=tauc11-tauc22;
N2c=tauc22-tauc33;
aijkl12=((27.0.*(a11.*a22.*(1-a11-a22)-(a12.^2).*(1-a11-a22))).*(-1.0/35.0+1.0/7.0.*(a11+a22))+(a12.^2).*(1-
27.*(a11.*a22.*(1-a11-a22))+27.0.*(a12.^2).*(1-a11-a22)));

export=[time tauc12];

pred=Gp(ref)*0.5*sin(freq(ref)*tspan)+Gdp(ref)*0.5*cos(freq(ref)*tspan);

export=[time tauc12 pred'];

subplot(1,3,2)
if c==4
   plot(tspan,pred,tspan,total12)
else
   plot(tspan,pred,tspan,tauc12)
end
```

```
exp=['frequency= ' num2str(w)];
legend('Gprime Gdoubleprime','stress')
title(exp)
   grid on
   grid minor
hold on

subplot(1,3,3)
plot(tspan,a11,tspan,a12,tspan,a22)
legend('a11','a12','a22')
title('Orientation Evolution')
   grid on
   grid minor

   function dyo = modepolymersub(t,y)

      tp11=y(1);
      tp12=y(2);
      tp22=y(3);
      tp33=y(4);
      a11=y(5);
      a12=y(6);
      a22=y(7);
      a33=1-a11-a22;
      dyo=zeros(7,1);

      dyo(1,1)=-alpha(x)*(tp11^2+tp12^2)/etap(x)-sigma*tp11/lambda(x)+2*r0*w*cos(w*t)*tp12-3*(1-
sigma)*(tp11*a11+tp12*a12)/lambda(x);
      dyo(2,1)=-alpha(x)*(tp11*tp12+tp12*tp22)/etap(x)-
sigma*tp12/lambda(x)+etap(x)*r0*w*cos(w*t)/lambda(x)+r0*w*cos(w*t)*tp22-3*(1-
sigma)/2/lambda(x)*(a11*tp12+a12*tp22+a12*tp11+a22*tp12);
      dyo(3,1)=-alpha(x)*(tp12^2+tp22^2)/etap(x)-sigma*tp22/lambda(x)-3*(1-
sigma)/lambda(x)*(tp12*a12+tp22*a22);
      dyo(4,1)=-alpha(x)*(tp33^2)/etap(x)-tp33*sigma/lambda(x)-3*(1-sigma)/lambda(x)*a33*tp33;

      dyo(5,1)=r0*w*cos(w*t)*a12+2*CI*abs(r0*w*cos(w*t))*(1.0-3.0*a11)+chi*r0*w*cos(w*t)*a12-...
         2*chi*r0*w*cos(w*t)*(3.0/7.0*a12*(27.0*a11*a22*(1-a11-a22)...
         -27.0*(a12^2)*(1-a11-a22))+(1.0-27.0*a11*a22*(1-a11-a22)+...
         27.0*(a12^2)*(1-a11-a22))*a11*a12);
      dyo(6,1)=1.0/2.0*r0*w*cos(w*t)*a22-1.0/2.0*r0*w*cos(w*t)*a11+chi*((1.0/2.0*r0*w*cos(w*t)*a22+...
         1.0/2.0*r0*w*cos(w*t)*a11)-(2.0*r0*w*cos(w*t))*((27.0*a11*a22*(1-a11-a22)-...
         27.0*(a12^2)*(1-a11-a22))*(-1.0/35.0+1.0/7.0*a11+1.0/7.0*a22)+...
         (1.0-27.0*a11*a22*(1-a11-a22)...
         +27.0*(a12^2)*(1-a11-a22))*a12^2))-6.0*CI*abs(r0*w*cos(w*t))*a12;
      dyo(7,1)=-r0*w*cos(w*t)*a12+2*CI*abs(r0*w*cos(w*t))*(1.0-3.0*a22)+chi*r0*w*cos(w*t)*a12-...
         2*chi*r0*w*cos(w*t)*(3.0/7.0*a12*(27.0*a11*a22*(1-a11-a22)-...
         27.0*(a12^2)*(1-a11-a22))+(1.0-27.0*a11*a22*(1-a11-a22)+...
         27.0*(a12^2)*(1-a11-a22))*a12*a22);

   end

end
```

Appendix H: MATLAB program for solving for G' and G" values based on the predicted stress wave predicted by the model for small amplitude oscillatory shear (SAOS) flows.

```
function GpGdp = SAOSFittingGpGdp
%This program models the stress wave from a small amplitude oscillatory
%shear flow and fits the optimum values of G' and G" to this stress wave

format long

Gpo=[55.6626    118.492 254.715 503.824 908.168 1605.89 2780.195    4406.575    6593.61 10148.9 14452.5
20191.5 26905.4 34823.55    44354.4 55135.45    67152.35    80926.2 95075.9 110655];
Gdpo=[618.6515  916.983 1479.19 2195.635    3208.705    4568.785    6359.81 8571.75 11351.8 14440.1 18048.75
21982.25    26022.35    30520.1 35164.85    39549.5 44375.45    49176.75    54481    60832];

freq=[0.01585   0.02512 0.03981 0.0631  0.1 0.15849 0.25119 0.39811 0.63096 1   1.58489 2.51189 3.98107
6.30957 10  15.8489 25.1189 39.8107 63.0957 100];

guess=[35.06125 118.77715   218.069 403.069 731.4956667 1295.72 2281.803333 3723.183333 5670.693333
8656.983333 12385.93333 17431.8 23216.63333 30205.2 38639.13333 47962.26667 58520.46667 70267.56667
82679.76667 96278.46667;
   532.1383333 799.4886667 1259.376667 1859.03 2733.77 3907.66 5450.736667 7393.43 9767.963333
12548.53333 15681.63333 19139.5 22729.5 26576.8 30379.83333 34434.86667 38387.86667 42793.26667 47297.4
52348.03333];

LB=zeros(2,1);

UB=[];
wt=0.0;   %in wt fraction

sigma=0.84;
CI=0.036;



for x=1:length(freq)

   x
   [stresst tspant]=SAOSSimulGsfit(freq(x),wt,sigma,CI);

   temp=find(tspant>=4*pi/freq(x),1);

   stress{x}=stresst(temp:end);
   tspan{x}=tspant(temp:end);



%options=optimset('Algorithm','interior-point');

[parameters fval exitflag] = fmincon(@Fitting,[guess(1,x) guess(2,x)],[],[],[],[],LB,UB,[])%,options)
```

```
Gpf(x)=parameters(1);
Gdpf(x)=parameters(2);
end

Gp
Gdp

predfinal=Gpf(x)*0.5*sin(freq(x)*tspan{x})+Gdpf(x)*0.5*cos(freq(x)*tspan{x});
predo=Gpo(x)*0.5*sin(freq(x)*tspan{x})+Gdpo(x)*0.5*cos(freq(x)*tspan{x});


plot(tspan{x},predfinal,tspan{x},stress{x},tspan{x},predo)
legend('Gprime Gdoubleprime','stress','original Gs')

GpGdp=[Gpf' Gdpf'];

finaler=sum(er)

function error = Fitting(para)

Gp=para(1)
Gdp=para(2)

error=0;

 pred=Gp*0.5*sin(freq(x)*tspan{x})+Gdp*0.5*cos(freq(x)*tspan{x});
   for i=1:length(tspan{x})
      error= error + (stress{x}(i)-pred(i))^2;
   end
   er(x)=error;

end

end




function [tauc12,tspan] = SAOSSimulGsfit(w,mass,sigma,CI)

x=1;

r0=0.5; %percent

if mass==0
   sigma=1;
end

re=33.0; %aspect ratio

chi=1.0*(re^2-1)/(re^2+1); %another form of aspect ratio
rf=1750.0; %fiber density
```

```
rs=1000.0; %polymer density

etap=[1100.046682   2920.193962 9277.693362 14316.47647 7408.914859]; %Final Parameters after pure polymer fitting to SAOS and transient
lambda=[0.012287706 0.11728325  0.743829501 2.689760868 15.39198862];
alpha=[0.50413207   0.678743122 0.680016347 0.317377081 0.27975612];

modes=5;

tspan=0:1/6/w:8*pi/w;

phi=1.0*rs*mass/(rf+(rs-rf)*mass);  %volume fraction
Ap=1.0*re^2/(3*log(sqrt(1.0*pi/phi)));  %A2, shape factor
%Ap=16*re^2/(3*log(1/phi))*(1-(log(log(1/phi))/log(1/phi))+0.6334/log(1/phi))  %A3
%Ap=16*re^2/(3*(log(1/phi)+log(log(1/phi))+1.4389)) %A4

orientation=[0.333, 0.0, 0.333];

tau_final=zeros(3,modes);

for x=1:modes
   initial=[0 0 0 0 orientation];
   [time, yo]=ode23tb(@modepolymersub,tspan,initial);

   L=length(time);

   if x==1
      tau1=zeros(L,modes);
      tau12=zeros(L,modes);
      tau2=zeros(L,modes);
      tau3=zeros(L,modes);
   end

   tau1(:,x)=yo(:,1);
   tau12(:,x)=yo(:,2);
   tau2(:,x)=yo(:,3);
   tau3(:,x)=yo(:,4);

   r=r0*w*cos(w*time);

   tau_final(:,x) = [tau1(L,x) tau12(L,x) tau2(L,x)]';

end

total11=sum(tau12,2);
total12=sum(tau12,2);
total22=sum(tau12,2);
total33=sum(tau12,2);

a11=yo(:,5);
a12=yo(:,6);
a22=yo(:,7);
a33=1-a11-a22;
```

```
eta=total12./r;
coef=2.0*eta*phi;

if r~=0
   disp('r~=0')
   tf12=(coef.*Ap.*r).*((27.0.*(a11.*a22.*(1-a11-a22)-(a12.^2).*(1-a11-a22))).*(-
1.0/35.0+1.0/7.0.*(a11+a22))+(a12.^2).*(1-27.*(a11.*a22.*(1-a11-a22))+27.0.*(a12.^2).*(1-a11-a22)));
   tf11=(coef.*Ap.*r).*((27.0.*(a11.*a22.*(1-a11-a22)-(a12.^2).*(1-a11-a22))).*(3.0/7.0.*(a12))+(a11.*a12).*(1-
27*(a11.*a22.*(1-a11-a22))+27.0.*(a12.^2).*(1-a11-a22)));
   tf22=(coef.*Ap.*r).*((27.0.*(a11.*a22.*(1-a11-a22)-(a12.^2).*(1-a11-a22))).*(3.0/7.0.*(a12))+(a22.*a12).*(1-
27*(a11.*a22.*(1-a11-a22))+27.0.*(a12.^2).*(1-a11-a22)));
   tf33=(coef.*Ap.*r).*((27.0.*(a11.*a22.*(1-a11-a22)-(a12.^2).*(1-a11-a22))).*(1.0/7.0.*(a12))+((1-a11-
a22).*a12).*(1-27.*(a11.*a22.*(1-a11-a22))+27.0.*(a12.^2).*(1-a11-a22)));
else
   disp('r==0')
   tf12=(2.0*phi.*total12.*Ap).*((27.0*(a11.*a22.*(1-a11-a22)-(a12.^2).*(1-a11-a22))).*(-
1.0/35.0+1.0/7.0*(a11+a22))+(a12.^2).*(1-27*(a11.*a22.*(1-a11-a22))+27.0*(a12.^2).*(1-a11-a22)));
   tf11=(2.0*phi.*total12.*Ap).*((27.0*(a11.*a22.*(1-a11-a22)-(a12.^2).*(1-a11-
a22))).*(3.0/7.0*(a12))+(a11.*a12).*(1-27*(a11.*a22.*(1-a11-a22))+27.0*(a12.^2).*(1-a11-a22)));
   tf22=(2.0*phi.*total12.*Ap).*((27.0*(a11.*a22.*(1-a11-a22)-(a12.^2).*(1-a11-
a22))).*(3.0/7.0*(a12))+(a22.*a12).*(1-27*(a11.*a22.*(1-a11-a22))+27.0*(a12.^2).*(1-a11-a22)));
   tf33=(2.0*phi.*total12.*Ap).*((27.0*(a11.*a22.*(1-a11-a22)-(a12.^2).*(1-a11-a22))).*(1.0/7.0*(a12))+((1-a11-
a22).*a12).*(1-27*(a11.*a22.*(1-a11-a22))+27.0*(a12.^2).*(1-a11-a22)));
end

tauc12=tf12+total12;
tauc11=tf11+total11;
tauc22=tf22+total22;
tauc33=tf33+total33;
etac=tauc12./r;
N1c=tauc11-tauc22;
N2c=tauc22-tauc33;
aijkl12=((27.0.*(a11.*a22.*(1-a11-a22)-(a12.^2).*(1-a11-a22))).*(-1.0/35.0+1.0/7.0.*(a11+a22))+(a12.^2).*(1-
27.*(a11.*a22.*(1-a11-a22))+27.0.*(a12.^2).*(1-a11-a22)));

   function dyo = modepolymersub(t,y)

      tp11=y(1);
      tp12=y(2);
      tp22=y(3);
      tp33=y(4);
      a11=y(5);
      a12=y(6);
      a22=y(7);
      a33=1-a11-a22;
      dyo=zeros(7,1);

      dyo(1,1)=-alpha(x)*(tp11^2+tp12^2)/etap(x)-sigma*tp11/lambda(x)+2*r0*w*cos(w*t)*tp12-3*(1-
sigma)*(tp11*a11+tp12*a12)/lambda(x);
      dyo(2,1)=-alpha(x)*(tp11*tp12+tp12*tp22)/etap(x)-
sigma*tp12/lambda(x)+etap(x)*r0*w*cos(w*t)/lambda(x)+r0*w*cos(w*t)*tp22-3*(1-
sigma)/2/lambda(x)*(a11*tp12+a12*tp22+a12*tp11+a22*tp12);
      dyo(3,1)=-alpha(x)*(tp12^2+tp22^2)/etap(x)-sigma*tp22/lambda(x)-3*(1-
sigma)/lambda(x)*(tp12*a12+tp22*a22);
```

```
    dyo(4,1)=-alpha(x)*(tp33^2)/etap(x)-tp33*sigma/lambda(x)-3*(1-sigma)/lambda(x)*a33*tp33;

    dyo(5,1)=r0*w*cos(w*t)*a12+2*CI*abs(r0*w*cos(w*t))*(1.0-3.0*a11)+chi*r0*w*cos(w*t)*a12-...
       2*chi*r0*w*cos(w*t)*(3.0/7.0*a12*(27.0*a11*a22*(1-a11-a22)...
       -27.0*(a12^2)*(1-a11-a22))+(1.0-27.0*a11*a22*(1-a11-a22)+...
       27.0*(a12^2)*(1-a11-a22))*a11*a12);
    dyo(6,1)=1.0/2.0*r0*w*cos(w*t)*a22-1.0/2.0*r0*w*cos(w*t)*a11+chi*((1.0/2.0*r0*w*cos(w*t)*a22+...
       1.0/2.0*r0*w*cos(w*t)*a11)-(2.0*r0*w*cos(w*t))*((27.0*a11*a22*(1-a11-a22)-...
       27.0*(a12^2)*(1-a11-a22))*(-1.0/35.0+1.0/7.0*a11+1.0/7.0*a22)+...
       (1.0-27.0*a11*a22*(1-a11-a22)...
       +27.0*(a12^2)*(1-a11-a22))*a12^2))-6.0*CI*abs(r0*w*cos(w*t))*a12;
    dyo(7,1)=-r0*w*cos(w*t)*a12+2*CI*abs(r0*w*cos(w*t))*(1.0-3.0*a22)+chi*r0*w*cos(w*t)*a12-...
       2*chi*r0*w*cos(w*t)*(3.0/7.0*a12*(27.0*a11*a22*(1-a11-a22)-...
       27.0*(a12^2)*(1-a11-a22))+(1.0-27.0*a11*a22*(1-a11-a22)+...
       27.0*(a12^2)*(1-a11-a22))*a12*a22);

  end

end
```